NAME

X – a portable, network-transparent window system

SYNOPSIS

The X Window System is a network transparent window system developed at MIT which runs on a wide range of computing and graphics machines. It should be relatively straightforward to build the MIT software distribution on most ANSI C and POSIX compliant systems. Commercial implementations are also available for a wide range of platforms.

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

X Window System is a trademark of the Massachusetts Institute of Technology.

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.

The number of programs that use *X* is quite large. Programs provided in the core MIT distribution include: a terminal emulator (*xterm*), a window manager (*twm*), a display manager (*xdm*), a console redirect program (*xconsole*), mail managing utilities (*xmh* and *xbiff*), a manual page browser (*xman*), a bitmap editor (*bitmap*), a resource editor (*editres*), a ditroff previewer (*xditview*), access control programs (*xauth* and *xhost*), user preference setting programs (*xrdb*, *xcmsdb*, *xset*, *xsetroot*, *xstdcmap*, and *xmodmap*), a load monitor (*xload*), clocks (*xclock* and *oclock*), a font displayer (*xfd*), utilities for listing information about fonts, windows, and displays (*xlsfonts*, *xfontsel*, *xwininfo*, *xlsclients*, *xdpyinfo*, and *xprop*), a diagnostic for seeing what events are generated and when (*xev*), screen image manipulation utilities (*xwd*, *xwud*, *xpr*, and *xmag*), and various demos (*xeyes*, *ico*, *xgc*, *x11perf*, etc.).

Many other utilities, window managers, games, toolkits, etc. are included as user-contributed software in the MIT distribution, or are available using anonymous ftp on the Internet. See your site administrator for details.

STARTING UP

There are two main ways of getting the X server and an initial set of client applications started. The particular method used depends on what operating system you are running and on whether or not you use other window systems in addition to X.

xdm **(the X Display Manager)**

If you want to always have X running on your display, your site administrator can set your machine up to use the X Display Manager *xdm*. This program is typically started by the system at boot time and takes care of keeping the server running and getting users logged in. If you are running *xdm*, you will see a window on the screen welcoming you to the system and asking for your username and password. Simply type them in as you would at a normal terminal, pressing the Return key after each. If you make a mistake, *xdm* will display an error message and ask you to try again. After you have successfully logged in, *xdm* will start up your X environment. By default, if you have an executable file named *.xsession* in your home directory, *xdm* will treat it as a program (or shell script) to run to start up your initial clients (such as terminal emulators, clocks, a window manager, user settings for things like the background, the speed of the pointer, etc.). Your site administrator can provide details.

xinit **(run manually from the shell)**

Sites that support more than one window system might choose to use the *xinit* program for starting X manually. If this is true for your machine, your site administrator will probably have provided a program named "x11", "startx", or "xstart" that will do site-specific initialization (such as loading convenient default resources, running a window manager, displaying a clock, and starting several terminal emulators) in a nice way. If not, you can build such a script using the *xinit* program. This utility simply runs one user-specified program to start the server, runs another to start up any desired clients, and then waits for either to finish. Since either or both of the user-specified programs may be a shell script, this gives substantial flexibility at the expense of a nice interface. For this reason, *xinit* is not intended for end users.

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* 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. For example,

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

The *xon* script can be used to start an X program on a remote machine; it automatically sets the DISPLAY variable correctly.

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. X servers generally 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*.

ACCESS CONTROL

An X server can use several types of access control. Mechanisms provided in Release 5 are:

Host Access	Simple host-based access control.
MIT-MAGIC-COOKIE-1	Shared plain-text "cookies".
XDM-AUTHORIZATION-1	Secure DES based private-keys.
SUN-DES-1	Based on Sun's secure rpc system.

Xdm initializes access control for the server, and also places authorization information 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*), the server no longer performs any authorization on connections from those machines. Be careful with this.

The file from which *Xlib* extracts authorization data 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.

If you use several machines, and share a common home directory across all of the machines by means of a network file system, then you never really have to worry about authorization files, the system should work correctly by default. Otherwise, as the authorization files are machine-independent, you can simply copy the files to share them. To manage authorization files, use *xauth*. 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, if the remote machine does not share a common home directory with your local machine. Note that authorization information transmitted ‘‘in the clear’’ through a network file system or using *ftp* or *rcp* 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 authorization data to know if this is actually a problem.

For more information on access control, see the *Xsecurity* manual page.

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. The MIT distribution comes with a window manager named *twm* which supports overlapping windows, popup menus, point-and-click or click-to-type input models, title bars, nice icons (and an icon manager for those who don't like separate icon windows).

See the user-contributed software in the MIT distribution for other popular window managers.

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). Similarly, collections of fonts that are based on a common type face (the variations are usually called roman, bold, italic, bold italic, oblique, and bold oblique) are called *families*.

Fonts come in various sizes. The X server supports *scalable* fonts, meaning it is possible to create a font of arbitrary size from a single source for the font. The server supports scaling from *outline* fonts and *bitmap* fonts. Scaling from outline fonts usually produces significantly better results than scaling from bitmap fonts.

An X server can obtain fonts from individual files stored in directories in the file system, or from one or more font servers, or from a mixtures of directories and font servers. The list of places the server looks when trying to find a font is controlled by its *font path*. Although most installations will choose to have the server start up with all of the commonly used font directories in the font path, the font path can be changed at any time with the *xset* program. However, it is important to remember that the directory names are on the **server**'s machine, not on the application's. The most common fonts use by X servers and font servers can be found in four directories:

/usr/lib/X11/fonts/misc
: This directory contains many miscellaneous bitmap fonts that are useful on all systems. It contains a family of fixed-width fonts, a family of fixed-width fonts from Dale Schumacher, several Kana fonts from Sony Corporation, two JIS Kanji fonts, two Hangul fonts from Daewoo Electronics, two Hebrew fonts from Joseph Friedman, the standard cursor font, two cursor fonts from Digital Equipment Corporation, and cursor and glyph fonts from Sun Microsystems. It also has various font name aliases for the fonts, including **fixed** and **variable**.

/usr/lib/X11/fonts/Speedo
: This directory contains outline fonts for Bitstream's Speedo rasterizer. A single font face, in normal, bold, italic, and bold italic, is provided, contributed by Bitstream, Inc.

/usr/lib/X11/fonts/75dpi
: This directory contains bitmap fonts contributed by Adobe Systems, Inc., Digital Equipment Corporation, Bitstream, Inc., Bigelow and Holmes, and Sun Microsystems, Inc. for 75 dots per inch displays. An integrated selection of sizes, styles, and weights are provided for each family.

/usr/lib/X11/fonts/100dpi
: This directory contains 100 dots per inch versions of some of the fonts in the *75dpi* directory.

Bitmap font files are usually created by compiling a textual font description into binary form, using *bdftopcf*. Font databases are created by running the *mkfontdir* program in the directory containing the source or 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.pcf ~/myfonts
% mkfontdir ~/myfonts
% xset fp rehash
```

The *xfontsel* and *xlsfonts* programs can be used to browse through the fonts available on a server. Font names tend to be fairly long as they contain all of the information needed to uniquely identify individual fonts. However, the X server supports wildcarding of font names, so the full specification

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

might be abbreviated as:

--courier-medium-r-normal--*-100-*-*-*-*-iso8859-1*

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

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

The *xlsfonts* program can be used to list all of the fonts that match a given pattern. With no arguments, it lists all available fonts. This will usually list the same font at many different sizes. To see just the base scalable font names, try using one of the following patterns:

```
-*-*-*-*-*-*-0-0-0-0-*-0-*-*
-*-*-*-*-*-*-0-0-75-75-*-0-*-*
-*-*-*-*-*-*-0-0-100-100-*-0-*-*
```

To convert one of the resulting names into a font at a specific size, replace one of the first two zeros with a nonzero value. The field containing the first zero is for the pixel size; replace it with a specific height in pixels to name a font at that size. Alternatively, the field containing the second zero is for the point size; replace it with a specific size in decipoints (there are 722.7 decipoints to the inch) to name a font at that size. The last zero is an average width field, measured in tenths of pixels; some servers will anamorphically scale if this value is specified.

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. A color can be specified either by an abstract color name, or by a numerical color specification. The numerical specification can identify a color in either device-dependent (RGB) or device-independent terms. Color strings are case-insensitive.

X supports the use of abstract color names, for example, "red", "blue". A value for this abstract name is obtained by searching one or more color name databases. *Xlib* first searches zero or more client-side databases; the number, location, and content of these databases is implementation dependent. If the name is not found, the color is looked up in the X server's database. The text form of this database is commonly stored in the file */usr/lib/X11/rgb.txt*.

A numerical color specification consists of a color space name and a set of values in the following syntax:

<color_space_name>:*<value>/.../<value>*

An RGB Device specification is identified by the prefix "rgb:" and has the following syntax:

rgb:*<red>/<green>/<blue>*

<red>, *<green>*, *<blue>* := *h* | *hh* | *hhh* | *hhhh*
h := single hexadecimal digits

Note that *h* indicates the value scaled in 4 bits, *hh* the value scaled in 8 bits, *hhh* the value scaled in 12 bits, and *hhhh* the value scaled in 16 bits, respectively. These values are passed directly to the X server, and are assumed to be gamma corrected.

The eight primary colors can be represented as:

black	rgb:0/0/0
red	rgb:ffff/0/0
green	rgb:0/ffff/0
blue	rgb:0/0/ffff
yellow	rgb:ffff/ffff/0
magenta	rgb:ffff/0/ffff
cyan	rgb:0/ffff/ffff
white	rgb:ffff/ffff/ffff

For backward compatibility, an older syntax for RGB Device is supported, but its continued use is not encouraged. The syntax is an initial sharp sign character followed by a numeric specification, in one of the following formats:

#RGB	(4 bits each)
#RRGGBB	(8 bits each)
#RRRGGGBBB	(12 bits each)
#RRRRGGGGBBBB	(16 bits each)

The R, G, and B represent single hexadecimal digits. When fewer than 16 bits each are specified, they represent the most-significant bits of the value (unlike the "rgb:" syntax, in which values are scaled). For example, #3a7 is the same as #3000a0007000.

An RGB intensity specification is identified by the prefix "rgbi:" and has the following syntax:

rgbi:*<red>*/*<green>*/*<blue>*

The red, green, and blue are floating point values between 0.0 and 1.0, inclusive. They represent linear intensity values, with 1.0 indicating full intensity, 0.5 half intensity, and so on. These values will be gamma corrected by *Xlib* before being sent to the X server. The input format for these values is an optional sign, a string of numbers possibly containing a decimal point, and an optional exponent field containing an E or e followed by a possibly signed integer string.

The standard device-independent string specifications have the following syntax:

CIEXYZ:*<X>*/*<Y>*/*<Z>*	(*none*, 1, *none*)
CIEuvY:*<u>*/*<v>*/*<Y>*	(~.6, ~.6, 1)
CIExyY:*<x>*/*<y>*/*<Y>*	(~.75, ~.85, 1)
CIELab:*<L>*/*<a>*/*<b>*	(100, *none*, *none*)
CIELuv:*<L>*/*<u>*/*<v>*	(100, *none*, *none*)
TekHVC:*<H>*/*<V>*/*<C>*	(360, 100, 100)

All of the values (C, H, V, X, Y, Z, a, b, u, v, y, x) are floating point values. Some of the values are constrained to be between zero and some upper bound; the upper bounds are given in parentheses above. The syntax for these values is an optional '+' or '-' sign, a string of digits possibly containing a decimal point, and an optional exponent field consisting of an 'E' or 'e' followed by an optional '+' or '-' followed by a string of digits.

For more information on device independent color, see the *Xlib* reference manual.

KEYBOARDS

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 control 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. The exact semantics of how the table is interpreted to produce keysyms depends on the particular program, libraries, and language input method used, but the following conventions for the first four keysyms in each row are generally adhered to:

The first four elements of the list are split into two groups of keysyms. Group 1 contains the first and second keysyms; Group 2 contains the third and fourth keysyms. Within each group, if the first element is alphabetic and the the second element is the special keysym *NoSymbol*, then the group is treated as equivalent to a group in which the first element is the lowercase letter and the second element is the uppercase letter.

Switching between groups is controlled by the keysym named MODE SWITCH, by attaching that keysym to some key and attaching that key to any one of the modifiers Mod1 through Mod5. This modifier is called the ``group modifier.'' Group 1 is used when the group modifier is off, and Group 2 is used when the group modifier is on.

Within a group, the modifier state determines which keysym to use. The first keysym is used when the Shift and Lock modifiers are off. The second keysym is used when the Shift modifier is on, when the Lock modifier is on and the second keysym is uppercase alphabetic, or when the Lock modifier is on and is interpreted as ShiftLock. Otherwise, when the Lock modifier is on and is interpreted as CapsLock, the state of the Shift modifier is applied first to select a keysym; but if that keysym is lowercase alphabetic, then the corresponding uppercase keysym is used instead.

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 provides a mechanism for storing default values for program resources (e.g. background color, window title, etc.) Resources are specified as strings that are read in from various places when an application is run. Program components are named in a hierarchical fashion, with each node in the hierarchy identified by a class and an instance name. At the top level is the class and instance name of the application itself. By convention, the class name of the application 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 | IncludeFile | ResourceSpec | <empty line>
Comment	=	"!" {<any character except null or newline>}
IncludeFile	=	"#" WhiteSpace "include" WhiteSpace FileName WhiteSpace
FileName	=	<valid filename for operating system>
ResourceSpec	=	WhiteSpace ResourceName WhiteSpace ":" WhiteSpace Value
ResourceName	=	[Binding] {Component Binding} ComponentName
Binding	=	"." | "*"
WhiteSpace	=	{<space> | <horizontal tab>}
Component	=	"?" | ComponentName
ComponentName	=	NameChar {NameChar}
NameChar	=	"a"–"z" | "A"–"Z" | "0"–"9" | "_" | "-"
Value	=	{<any character except null or unescaped newline>}

Elements separated by vertical bar (|) are alternatives. Curly braces ({...}) indicate zero or more repetitions of the enclosed elements. Square brackets ([...]) indicate that the enclosed element is optional. Quotes ("...") are used around literal characters.

IncludeFile lines are interpreted by replacing the line with the contents of the specified file. The word "include" must be in lowercase. The filename is interpreted relative to the directory of the file in which the line occurs (for example, if the filename contains no directory or contains a relative directory specification).

If a ResourceName contains a contiguous sequence of two or more Binding characters, the sequence will be replaced with single "." character if the sequence contains only "." characters, otherwise the sequence will be replaced with a single "*" character.

A resource database never contains more than one entry for a given ResourceName. If a resource file contains multiple lines with the same ResourceName, the last line in the file is used.

Any whitespace character before or after the name or colon in a ResourceSpec are ignored. To allow a Value to begin with whitespace, the two-character sequence ''*space*'' (backslash followed by space) is recognized and replaced by a space character, and the two-character sequence ''*tab*'' (backslash followed by horizontal tab) is recognized and replaced by a horizontal tab character. To allow a Value to contain embedded newline characters, the two-character sequence ''\n'' is recognized and replaced by a newline character. To allow a Value to be broken across multiple lines in a text file, the two-character sequence ''*newline*'' (backslash followed by newline) is recognized and removed from the value. To allow a Value to contain arbitrary character codes, the four-character sequence ''*nnn*'', where each *n* is a digit character in the range of ''0''–''7'', is recognized and replaced with a single byte that contains the octal value specified by the sequence. Finally, the two-character sequence ‘‘\\’’ is recognized and replaced with a single backslash.

When an application looks for the value of a resource, it specifies a complete path in the hierarchy, with both class and instance names. However, resource values are usually given with only partially specified names and classes, using pattern matching constructs. An asterisk (*) is a loose binding and is used to represent any number of intervening components, including none. A period (.) is a tight binding and is used to separate immediately adjacent components. A question mark (?) is used to match any single component name or class. A database entry cannot end in a loose binding; the final component (which cannot be "?") must be specified. The lookup algorithm searches the resource database for the entry that most closely matches (is most specific for) the full name and class being queried. When more than one database entry matches the full name and class, precedence rules are used to select just one.

The full name and class are scanned from left to right (from highest level in the hierarchy to lowest), one component at a time. At each level, the corresponding component and/or binding of each matching entry is determined, and these matching components and bindings are compared according to precedence rules. Each of the rules is applied at each level, before moving to the next level, until a rule selects a single entry over all others. The rules (in order of precedence) are:

1. An entry that contains a matching component (whether name, class, or "?") takes precedence over entries that elide the level (that is, entries that match the level in a loose binding).

2. An entry with a matching name takes precedence over both entries with a matching class and entries that match using "?". An entry with a matching class takes precedence over entries that match using "?".

3. An entry preceded by a tight binding takes precedence over entries preceded by a loose binding.

Programs based on the X Tookit Intrinsics obtain resources from the following sources (other programs usually support some subset of these 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 of the first screen using the *xrdb* program. This is frequently taken care of when the user starts up X through the display manager or *xinit*.

SCREEN_RESOURCES root window property
Any resources specific to a given screen (e.g. colors) that should be available to clients on all machines should be stored in the SCREEN_RESOURCES property on the root window of that screen. The *xrdb* program will sort resources automatically and place them in RESOURCE_MANAGER or SCREEN_RESOURCES, as appropriate.

application-specific files
Directories named by the environment variable XUSERFILESEARCHPATH or the environment variable XAPPLRESDIR, plus directories in a standard place (usually under /usr/lib/X11/, but this can be overridden with the XFILESEARCHPATH environment variable) are searched for for application-specific resources. For example, application default resources are usually kept in /usr/lib/X11/app-defaults/. See the *X Toolkit Intrinsics - C Language Interface* manual for details.

XENVIRONMENT
Any user- and machine-specific resources may be specified by setting the XENVIRONMENT environment variable to the name of a resource file to be loaded by all applications. If this variable is not defined, a file named *$HOME*/.Xdefaults-*hostname* is looked for instead, where *hostname* is the name of the host where the application is executing.

–xrm *resourcestring*
Resources can also be 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*cutToBeginningOfLine: 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:  rgb:5b/76/86
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 frequently how user-friendly startup scripts merge 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 *Resource Manager Functions* 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 $HOME/.Xresources
% xmodmap -e "keysym BackSpace = Delete"
% mkfontdir /usr/local/lib/X11/otherfonts
% xset fp+ /usr/local/lib/X11/otherfonts
% xmodmap $HOME/.keymap.km
% xsetroot -solid 'rgbi:.8/.8/.8'
% xset b 100 400 c 50 s 1800 r on
% xset q
% twm
% xmag
% xclock -geometry 48x48-0+0 -bg blue -fg white
% xeyes -geometry 48x48-48+0
% xbiff -update 20
% xlsfonts '*helvetica*'
% 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 $*
% xon filesysmachine xload
```

DIAGNOSTICS

A wide variety of error messages are generated from various programs. 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/lib/X11/XErrorDB*. If this file is not present, error messages will be rather terse and cryptic.

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 in the file that gets loaded onto the RESOURCE_MANAGER property using the *xrdb* 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

XConsortium(1), XStandards(1), Xsecurity(1), appres(1), auto_box(1), bdftopcf(1), beach_ball(1), bitmap(1), editres(1), fs(1), fsinfo(1), fslsfonts(1), fstobdf(1), ico(1), imake(1), listres(1), lndir(1), makedepend(1), maze(1), mkdirhier(1), mkfontdir(1), oclock(1), plbpex(1), puzzle(1), resize(1), showfont(1), showrgb(1), twm(1), viewres(1), x11perf(1), x11perfcomp(1), xauth(1), xbiff(1), xcalc(1), xclipboard(1), xclock(1), xcmsdb(1), xcmstest(1), xconsole(1), xcutsel(1), xditview(1), xdm(1), xdpr(1), xdpyinfo(1), xedit(1), xev(1), xeyes(1), xfd(1), xfontsel(1), xgas(1), xgc(1), xhost(1), xinit(1), xkill(1), xload(1), xlogo(1), xlsatoms(1), xlsclients(1), xlsfonts(1), xmag(1), xman(1), xmh(1), xmkmf(1), xmodmap(1), xon(1), xpr(1), xprop(1), xrdb(1), xrefresh(1), xset(1), xsetroot(1), xstdcmap(1), xterm(1), xwd(1), xwininfo(1), xwud(1), Xserver(1), Xdec(1), XmacII(1), Xmips(1), Xqdss(1), Xqvss(1), Xsun(1), X386(1), kbd_mode(1), *Xlib – C Language X Interface*, and *X Toolkit Intrinsics – C Language Interface*

COPYRIGHT

TRADEMARKS

X Window System is a trademark of MIT.

AUTHORS

A cast of thousands, literally. The MIT Release 5 distribution is brought to you by the MIT X Consortium. The names of all people who made it a reality will be found in the individual documents and source files. The staff members at MIT responsible for this release are: Donna Converse (MIT X Consortium), Stephen Gildea (MIT X Consortium), Susan Hardy (MIT X Consortium), Jay Hersh (MIT X Consortium), Keith Packard (MIT X Consortium), David Sternlicht (MIT X Consortium), Bob Scheifler (MIT X Consortium), and Ralph Swick (Digital/MIT Project Athena).

NAME

MIT X Consortium – MIT X Consortium information

SYNOPSIS

Release 5 of X Version 11 is brought to you by the MIT X Consortium.

DESCRIPTION

X Window System research began in the summer of 1984 at MIT as an informal joint undertaking between the Laboratory for Computer Science and Project Athena. Since that time, researchers and engineers at numerous other universities and companies have been involved in the design and implementation.

The MIT X Consortium was formed in January of 1988 to further the development of the X Window System. It is part of the Laboratory for Computer Science at MIT, and has as its major goal the promotion of cooperation within the computer industry in the creation of standard software interfaces at all layers in the X Window System environment. MIT's role is to provide the vendor-neutral architectural and administrative leadership required to make this work. The Consortium is financially self-supporting, with membership open to any organization. There are two categories of membership, Member (for large organizations) and Affiliate (for smaller organizations). A list of members as of the release of X11R5 is given below.

The Director of the X Consortium acts as the ultimate architect for all X specifications and software. The activities of the Consortium are overseen by an MIT Steering Committee, which helps set policy and provides strategic guidance and review of the Consortium's activities. An Advisory Committee made up of member representatives meets regularly to review the Consortium's plans, assess its progress, and suggest future directions.

Most of the Consortium's activities take place via electronic mail, with meetings when required. As designs and specifications take shape, interest groups are formed from experts in the participating organizations. Typically a small multi-organization architecture team leads the design, with others acting as close observers and reviewers. Once a complete specification is produced, it may be submitted for formal technical review by the Consortium as a proposed standard. The standards process includes public review (outside the Consortium) and a demonstration of proof of concept, typically established with a public, portable implementation of the specification, (although other methods are possible on a case by case basis).

The MIT staff of the Consortium also maintains a software and documentation collection, containing both Consortium-developed and user-contributed materials, and endeavors to make periodic distributions of this collection available to the public without license and for a minimal fee. X11R5 is the fifth such distribution. The Consortium also sponsors an annual conference, open to the public, to promote the exchange of technical information about X.

STAFF

The Director of the X Consortium is Bob Scheifler, rws@expo.lcs.mit.edu. The staff of the X Consortium are all full-time research staff employed by the MIT Laboratory for Computer Science. The staff members are as follows.

Donna Converse, converse@expo.lcs.mit.edu
Stephen Gildea, gildea@expo.lcs.mit.edu
Susan Hardy, sh@expo.lcs.mit.edu
Jay Hersh, hersh@expo.lcs.mit.edu
Keith Packard, keith@expo.lcs.mit.edu
Dave Sternlicht, dave@expo.lcs.mit.edu

Ralph Swick, an employee of Digital Equipment Corporation working at MIT Project Athena, also worked directly with the Consortium staff on this release. Former MIT staff members Jim Fulton and Chris Peterson contributed to the early development of the release.

POSTAL ADDRESS

The MIT X Consortium can be reached by writing to

Bob Scheifler
MIT X Consortium
Laboratory for Computer Science
545 Technology Square
Cambridge, MA 02139

MEMBERS

Apple Computer, Inc.
Bull S.A.
Control Data Corporation
Convex Computer Corporation
Cray Research, Inc.
Data General Corporation
Digital Equipment Corp.
E.I. DuPont de Nemours & Company, Inc.
Eastman Kodak Company
Evans & Sutherland
Fujitsu Limited
Hewlett-Packard Company
Hitachi Ltd.
Hughes Aircraft Company
IBM Corporation
Intergraph Corp.
Kubota Corp.
Matsushita Electric Industrical Co., Ltd.
MIPS Computer Systems
Mitsubishi Electric Corporation
Motorola, Inc.
NCR Corporation
NEC Corporation
Network Computing Devices
Nippon Telegraph and Telephone Corporation
Oki Electric Industry Co., Ltd.
Olivetti Systems & Networks
OMRON Corporation
Prime Computer, Inc.
The Santa Cruz Operation, Inc.
Sequent Computer Systems Inc.
Siemens Nixdorf Informationssysteme AG
Silicon Graphics, Inc.
Sony Corporation
Sun Microsystems, Inc.
Tandberg Data A/S
Tektronix, Inc.
Texas Instruments, Inc.
Unisys Corporation
Unix Systems Laboratories/American Telephone & Telegraph Company
Xerox Corporation

AFFILIATES

Adobe Systems Inc.
AGE Logic, Inc.
ASTEC, Inc.
Athenix Corporation
Bitstream, Inc.
CETIA - Compagnie Europeene des Techniques de l’Ingenierie Assistee
Chromatics, Inc.
Codonics, Inc.
Computer & Communication Research Laboratories
Frame Technology Corp.
Gfxbase, Inc.
GIPSI S.A.
HaL Computer Systems, Inc.
Institute for Information Industry
Interactive Systems Corporation
Integrated Computer Solutions, Inc.
IXI Limited
Japan Computer Corporation
Jupiter Systems
KAIST - Korean Advanced Institute of Science and Technology
Labtam Australia
Locus Computing Corporation
Metro Link, Inc.
University of Lowell
Megatek Corporation
Metheus Corporation
MITRE Corporation
Objectivity, Inc.
Open Software Foundation
O’Reilly & Associates, Inc.
PCS Computer Systeme GmbH
Peritek Corp.
PsiTech, Inc.
Quarterdeck Office Systems
Ramtek Corporation
Samsung Electronics Corp.
ShoGraphics, Inc.
Snitily Graphics Consulting Services
Solbourne Computer, Inc.
SOUM Corporation
SPARC International
Spectragraphics Corp.
Stanford University
Stardent Computer
Strategic Research Institute Inc.
Sumitomo Electric Workstation
Tatung Science and Technology
Template Graphics Software
Tyan Computer
Unipalm XTech
Visix Software, Inc.
Visual Information Technologies, Inc.

Visual Technology, Inc.
X/Open Company Ltd.

NAME

X Standards – MIT X Consortium Standards

SYNOPSIS

The major goal of the MIT X Consortium is to promote cooperation within the computer industry in the creation of standard software interfaces at all layers in the X Window System environment. The status of various standards and the software in the X11R5 distribution is explained below.

STANDARDS

The following documents are MIT X Consortium standards:

X Window System Protocol
X Version 11, Release 5
Robert W. Scheifler

Xlib - C Language X Interface
X Version 11, Release 5
James Gettys, Robert W. Scheifler, Ron Newman

X Toolkit Intrinsics - C Language Interface
X Version 11, Release 5
Joel McCormack, Paul Asente, Ralph R. Swick

Bitmap Distribution Format
Version 2.1
X Version 11, Release 5

Inter-Client Communication Conventions Manual
Version 1.1
X Version 11, Release 5
David Rosenthal

Compound Text Encoding
Version 1.1
X Version 11, Release 5
Robert W. Scheifler

X Logical Font Description Conventions
Version 1.4
X Version 11, Release 5
Jim Flowers

X Display Manager Control Protocol
Version 1.0
X Version 11, Release 5
Keith Packard

X11 Nonrectangular Window Shape Extension
Version 1.0
X Version 11, Release 5
Keith Packard

X11 Input Extension Protocol Specification
Version 1.0
X Version 11, Release 5

George Sachs, Mark Patrick

X11 Input Extension Library Specification
X Version 11, Release 5
Mark Patrick, George Sachs

The X Font Service Protocol
Version 1.0
X Version 11, Release 5
Jim Fulton

DRAFT STANDARDS

The following documents are currently draft standards of the MIT X Consortium.

PEX Protocol Specification
Version 5.0P
Sally Barry (editor)

Extending X for Double-Buffering, Multi-Buffering, and Stereo
Version 3.2
Jeffrey Friedberg, Larry Seiler, Jeff Vroom

INCLUDE FILES

The following include files are part of the Xlib standard.

<X11/cursorfont.h>
<X11/keysym.h>
<X11/keysymdef.h>
<X11/X.h>
<X11/Xatom.h>
<X11/Xcms.h>
<X11/Xlib.h>
<X11/Xlibint.h>
<X11/Xproto.h>
<X11/Xprotostr.h>
<X11/Xresource.h>
<X11/Xutil.h>
<X11/X10.h>

The following include files are part of the X Toolkit Intrinsics standard.

<X11/Composite.h>
<X11/CompositeP.h>
<X11/Constraint.h>
<X11/ConstrainP.h>
<X11/Core.h>
<X11/CoreP.h>
<X11/Intrinsic.h>
<X11/IntrinsicP.h>
<X11/Object.h>
<X11/ObjectP.h>
<X11/RectObj.h>
<X11/RectObjP.h>
<X11/Shell.h>

<X11/ShellP.h>
<X11/StringDefs.h>
<X11/Vendor.h>
<X11/VendorP.h>

The following include file is part of the Nonrectangular Window Shape Extension standard.

<X11/extensions/shape.h>

The following include files are part of the X Toolkit Intrinsics standard.

<X11/extensions/XI.h>
<X11/extensions/XInput.h>
<X11/extensions/XIproto.h>

The following include file is part of the Multi-Buffering draft standard.

<X11/extensions/multibuf.h>

NON STANDARDS

The X11R5 distribution contains *sample* implementations, not *reference* implementations. Although much of the code is believed to be correct, the code should be assumed to be in error wherever it conflicts with the specification.

At the public release of X11R5, the only MIT X Consortium standards are the ones listed above. No other documents, include files, or software in X11R5 carry special status within the X Consortium. For example, none of the following are standards: internal interfaces of the sample server; the MIT-SHM extension; the Input Synthesis extension; the Athena Widget Set; the Xmu library; the Xau library; CLX, the Common Lisp interface to X; the RGB database; the fonts distributed with X11R5; the applications distributed with X11R5; the include files <X11/XWDFile.h>, <X11/Xfuncproto.h>, <X11/Xfuncs.h>, <X11/Xosdefs.h>, and <X11/Xos.h>; the bitmap files in <X11/bitmaps>.

NAME

X Security − X display access control

SYNOPSIS

X provides mechanism for implementing many access control systems. Release 5 includes four mechanisms:

Host Access	Simple host-based access control.
MIT-MAGIC-COOKIE-1	Shared plain-text "cookies".
XDM-AUTHORIZATION-1	Secure DES based private-keys.
SUN-DES-1	Based on Sun's secure rpc system.

ACCESS SYSTEM DESCRIPTIONS

Host Access

Any client on a host in the host access control list is allowed access to the X server. This system can work reasonably well in an environment where everyone trusts everyone, or when only a single person can log in to a given machine, and is easy to use when the list of hosts used is small. This system does not work well when multiple people can log in to a single machine and mutual trust does not exist. The list of allowed hosts is stored in the X server and can be changed with the *xhost* command. When using the more secure mechanisms listed below, the host list is normally configured to be the empty list, so that only authorized programs can connect to the display.

MIT-MAGIC-COOKIE-1

When using MIT-MAGIC-COOKIE-1, the client sends a 128 bit "cookie" along with the connection setup information. If the cookie presented by the client matches one that the X server has, the connection is allowed access. The cookie is chosen so that it is hard to guess; *xdm* generates such cookies automatically when this form of access control is used. The user's copy of the cookie is usually stored in the *.Xauthority* file in the home directory, although the environment variable **XAUTHORITY** can be used to specify an alternate location. *Xdm* automatically passes a cookie to the server for each new login session, and stores the cookie in the user file at login.

The cookie is transmitted on the network without encryption, so there is nothing to prevent a network snooper from obtaining the data and using it to gain access to the X server. This system is useful in an environment where many users are running applications on the same machine and want to avoid interference from each other, with the caveat that this control is only as good as the access control to the physical network. In environments where network-level snooping is difficult, this system can work reasonably well.

XDM-AUTHORIZATION-1

For sites in the US, Release 5 contains a DES-based access control mechanism called XDM-AUTHORIZATION-1. It is similar in usage to MIT-MAGIC-COOKIE-1 in that a key is stored in the *.Xauthority* file and is shared with the X server. However, this key consists of two parts - a 56 bit DES encryption key and 64 bits of random data used as the authenticator.

When connecting to the X server, the application generates 192 bits of data by combining the current time in seconds (since 00:00 1/1/1970 GMT) along with 48 bits of "identifier". For TCP/IP connections, the identifier is the address plus port number; for local connections it is the process ID and 32 bits to form a unique id (in case multiple connections to the same server are made from a single process). This 192 bit packet is then encrypted using the DES key and sent to the X server, which is able to verify if the requestor is authorized to connect by decrypting with the same DES key and validating the authenticator and additional data. This system is useful in many environments where host-based access control is inappropriate and where network security cannot be ensured.

SUN-DES-1

Recent versions of SunOS (and some other systems) have included a secure public key remote procedure call system. This system is based on the notion of a network principal; a user name and NIS domain pair. Using this system, the X server can securely discover the actual user name of the requesting process. It involves encrypting data with the X servers public key, and so the identity of the user who started the X server is needed for this; this identity is stored in the *.Xauthority* file. By extending the semantics of "host address" to include this notion of network principal, this form of access control is very easy to use. To allow access by a new user, use *xhost*. For example,

xhost keith@ joe@mit.edu

adds "keith" from the NIS domain of the local machine, and "joe" in the "mit.edu" NIS domain. For keith or joe to successfully connect to the display, they must add the principal who started the server to their *.Xauthority* file. For example:

xauth add expo.lcs.mit.edu:0 SUN-DES-1 unix.expo.lcs.mit.edu:x.lcs.mit.edu

This system only works on machines which support Secure RPC, and only for users which have set up the appropriate public/private key pairs on their system. See the Secure RPC documentation for details.

THE AUTHORIZATION FILE

Except for Host Access control, each of these systems uses data stored in the *.Xauthority* file to generate the correct authorization information to pass along to the X server at connection setup. MIT-MAGIC-COOKIE-1 and XDM-AUTHORIZATION-1 store secret data in the file; so anyone who can read the file can gain access to the X server. SUN-DES-1 stores only the identity of the principal who started the server (unix.*hostname@domain* when the server is started by *xdm*), and so it is not useful to anyone not authorized to connect to the server.

Each entry in the *.Xauthority* file matches a certain connection family (TCP/IP, DECnet or local connections) and X display name (hostname plus display number). This allows multiple authorization entries for different displays to share the same data file. A special connection family (FamilyWild, value 65535) causes an entry to match every display, allowing the entry to be used for all connections. Each entry additionally contains the authorization name and whatever private authorization data is needed by that authorization type to generate the correct information at connection setup time.

The *xauth* program manipulates the *.Xauthority* file format. It understands the semantics of the connection families and address formats, displaying them in an easy to understand format. It also understands that SUN-DES-1 uses string values for the authorization data, and displays them appropriately.

The X server (when running on a workstation) reads authorization information from a file name passed on the command line with the *-auth* option (see the *Xserver* manual page). The authorization entries in the file are used to control access to the server. In each of the authorization schemes listed above, the data needed by the server to initialize an authorization scheme is identical to the data needed by the client to generate the appropriate authorization information, so the same file can be used by both processes. This is especially useful when *xinit* is used.

MIT-MAGIC-COOKIE-1

This system uses 128 bits of data shared between the user and the X server. Any collection of bits can be used. *Xdm* generates these keys using a cryptographically secure pseudo random number generator, and so the key to the next session cannot be computed from the current session key.

XDM-AUTHORIZATION-1

This system uses two pieces of information. First, 64 bits of random data, second a 56 bit DES encryption key (again, random data) stored in 8 bytes, the last byte of which is ignored. *Xdm* generates these keys using the same random number generator as is used for MIT-MAGIC-COOKIE-1.

SUN-DES-1

This system needs a string representation of the principal which identifies the associated X server. When *xdm* starts the X server, it uses the root principal for the machine on which it is running (unix.*hostname@domain*, e.g. "unix.expire.lcs.mit.edu@x.lcs.mit.edu"). Putting the correct principal name in the *.Xauthority* file causes Xlib to generate the appropriate authorization information using the secure RPC library.

FILES

.Xauthority

SEE ALSO

X(1), xdm(1), xauth(1), xhost(1), xinit(1), Xserver(1)

NAME

appres - list application resource database

SYNOPSIS

appres [[class [instance]] [-1] [toolkitoptions]

DESCRIPTION

The *appres* program prints the resources seen by an application (or subhierarchy of an application) with the specified *class* and *instance* names. It can be used to determine which resources a particular program will load. For example,

```
% appres XTerm
```

will list the resources that any *xterm* program will load. If no application class is specified, the class *-AppResTest-* is used.

To match a particular instance name, specify an instance name explicitly after the class name, or use the normal Xt toolkit option. For example,

```
% appres XTerm myxterm
```

or

```
% appres XTerm -name myxterm
```

To list resources that match a subhierarchy of an application, specify hierarchical class and instance names. The number of class and instance components must be equal, and the instance name should not be specified with a toolkit option. For example,

```
% appres Xman.TopLevelShell.Form xman.topBox.form
```

will list the resources of widgets of *xman* topBox hierarchy. To list just the resources matching a specific level in the hierarcy, use the –1 option. For example,

```
% appres XTerm.VT100 xterm.vt100 -1
```

will list the resources matching the *xterm* vt100 widget.

SEE ALSO

X(1), xrdb(1), listres(1)

COPYRIGHT

AUTHOR

Jim Fulton, MIT X Consortium

NAME

auto_box – display a rotating cube using PHIGS

SYNOPSIS

auto_box

DESCRIPTION

auto_box is a simple program that displays four views of a rotating cube. It initially displays the cube with filled polygons, then with hollow polygons, or polylines. Four views of the cube are presented in four quadrants. each with a different rotation applied. In the first and fourth quadrants the back faces and front faces are respectively culled out, demonstrating one of the PHIGS PLUS features. auto_box uses only PHIGS indexed colors. Type auto_box to run the program. It exits after all views and rotations are completed once.

SEE ALSO

PEX-SI User Guide

COPYRIGHT

NAME

bdftopcf – convert font from Bitmap Distribution Format to Portable Compiled Format

SYNOPSIS

bdftopcf [-option ...] font-file.bdf

DESCRIPTION

Bdftopcf is the release 5 font compiler. Fonts in Portable Compiled Format can be read by any architecture, although the file is structured to allow one particular architecture to read them directly without reformatting. This allows fast reading on the appropriate machine, but the files are still portable (but read more slowly) on other machines.

OPTIONS

–p*n* Sets the font glyph padding. Each glyph in the font will have each scanline padded in to a multiple of *n* bytes, where *n* is 1, 2, 4 or 8.

–u*n* Sets the font scanline unit. When the font bit order is different from the font byte order, the scanline unit *n* describes what unit of data (in bytes) are to be swapped; the unit *i* can be 1, 2 or 4 bytes.

–m Sets the font bit order to MSB (most significant bit) first. Bits for each glyph will be placed in this order; i.e. the left most bit on the screen will be in the highest valued bit in each unit.

–l Sets the font bit order to LSB (least significant bit) first. The left most bit on the screen will be in the lowest valued bit in each unit.

–M Sets the font byte order to MSB first. All multi-byte data in the file (metrics, bitmaps and everything else) will be written most significant byte first.

–L Sets the font byte order to LSB first. All multi-byte data in the file (metrics, bitmaps and everything else) will be written least significant byte first.

–t When this option is specified, *bdftopcf* will convert fonts into "terminal" fonts when possible. A terminal font has each glyph image padded to the same size; the X server can usually render these types of fonts more quickly.

–i This option inhibits the normal computation of ink metrics. When a font has glyph images which do not fill the bitmap image (i.e. the "on" pixels don’t extend to the edges of the metrics) *bdftopcf* computes the actual ink metrics and places them in the .pcf file; the –t option inhibits this behaviour.

–o <output-file-name>

By default *bdftopcf* writes the pcf file to standard output; this option gives the name of a file to be used instead.

SEE ALSO

X(1)

COPYRIGHT

AUTHOR

Keith Packard, MIT X Consortium

NAME

beach_ball – display a bouncing sphere using PHIGS

SYNOPSIS

beach_ball

DESCRIPTION

beach_ball is a simple program that displays a bouncing multicolored pseudo-sphere (a beach ball). It uses filled polygons for the faces of the beach ball. Type beach_ball to run this program. You must type Control-C to exit this program, as it will run in a continuous loop.

On server implementations that have not been optimized, it may be hard to visualize the bouncing motion of the ball. Because PEX requests may be buffered in queues, there may be a delay between typing Control-C and the client exiting.

SEE ALSO

PEX-SI User Guide

COPYRIGHT

NAME

bitmap, bmtoa, atobm – bitmap editor and converter utilities for the X Window System

SYNOPSIS

bitmap [–*options* . . .] [*filename*] [*basename*]

bmtoa [**–chars** . . .] [*filename*]

atobm [**–chars** *cc*] [**–name** *variable*] [**–xhot** *number*] [**–yhot** *number*] [*filename*]

DESCRIPTION

The *bitmap* program is a rudimentary tool for creating or editing rectangular images made up of 1's and 0's. Bitmaps are used in X for defining clipping regions, cursor shapes, icon shapes, and tile and stipple patterns.

The *bmtoa* and *atobm* filters convert *bitmap* files (FILE FORMAT) to and from ASCII strings. They are most commonly used to quickly print out bitmaps and to generate versions for including in text.

COMMAND LINE OPTIONS

Bitmap supports the standard X Toolkit command line arguments (see *X*(1)). The following additional arguments are supported as well.

–size *WIDTHxHEIGHT*
Specifies size of the grid in squares.

–sw *dimension*
Specifies the width of squares in pixels.

–sh *dimension*
Specifies the height of squares in pixels.

–gt *dimension*
Grid tolerance. If the square dimensions fall below the specified value, grid will be automatically turned off.

–grid, +grid
Turns on or off the grid lines.

–axes, +axes
Turns on or off the major axes.

–dashed, +dashed
Turns on or off dashing for the frame and grid lines.

–stippled, +stippled
Turns on or off stippling of highlighted squares.

–proportional, +proportional
Turns proportional mode on or off. If proportional mode is on, square width is equal to square height. If proportional mode is off, *bitmap* will use the smaller square dimension, if they were initially different.

–dashes *filename*
Specifies the bitmap to be used as a stipple for dashing.

–stipple *filename*
Specifies the bitmap to be used as a stipple for highlighting.

–hl *color*
Specifies the color used for highlighting.

–fr *color*
Specifies the color used for the frame and grid lines.

filename
Specifies the bitmap to be initially loaded into the program. If the file does not exist, *bitmap* will assume it is a new file.

basename
Specifies the basename to be used in the C code output file. If it is different than the basename in the working file, *bitmap* will change it when saving the file.

Bmtoa accepts the following option:

–chars *cc*
This option specifies the pair of characters to use in the string version of the bitmap. The first character is used for 0 bits and the second character is used for 1 bits. The default is to use dashes (–) for 0's and sharp signs (#) for 1's.

Atobm accepts the following options:

–chars *cc*
This option specifies the pair of characters to use when converting string bitmaps into arrays of numbers. The first character represents a 0 bit and the second character represents a 1 bit. The default is to use dashes (–) for 0's and sharp signs (#) for 1's.

–name *variable*
This option specifies the variable name to be used when writing out the bitmap file. The default is to use the basename of the *filename* command line argument or leave it blank if the standard input is read.

–xhot *number*
This option specifies the X coordinate of the hotspot. Only positive values are allowed. By default, no hotspot information is included.

–yhot *number*
This option specifies the Y coordinate of the hotspot. Only positive values are allowed. By default, no hotspot information is included.

USAGE

Bitmap displays grid in which each square represents a single bit in the picture being edited. Actual size of the bitmap image, as it would appear normaly and inverted, can be obtained by pressing **Meta-I** key. You are free to move the image popup out of the way to continue editing. Pressing the left mouse button in the popup window or **Meta-I** again will remove the real size bitmap image.

If the bitmap is to be used for defining a cursor, one of the squares in the images may be designated as the hot spot. This determines where the cursor is actually pointing. For cursors with sharp tips (such as arrows or fingers), this is usually at the end of the tip; for symmetric cursors (such as crosses or bullseyes), this is usually at the center.

Bitmaps are stored as small C code fragments suitable for including in applications. They provide an array of bits as well as symbolic constants giving the width, height, and hot spot (if specified) that may be used in creating cursors, icons, and tiles.

EDITING

To edit a bitmap image simply click on one of the buttons with drawing commands (**Point, Curve, Line, Rectangle,** etc.) and move the pointer into the bitmap grid window. Press one of the buttons on your mouse and the appropriate action will take place. You can either set, clear or invert the gird squares. Setting a grid square corresponds to setting a bit in the bitmap image to 1. Clearing a grid square corresponds to setting a bit in the bitmap image to 0. Inverting a grid square corresponds to changing a bit in the bitmap image from 0 to 1 or 1 to 0, depending what its previous state was. The default behavior of mouse buttons is as specified below.

MouseButton1	Set
MouseButton2	Invert

MouseButton3	Clear
MouseButton4	Clear
MouseButton5	Clear

This default behavior can be changed by setting the button function resources. An example is provided below.

```
bitmap*button1Function: Set
bitmap*button2Function: Clear
bitmap*button3Function: Invert
etc.
```

The button function applies to all drawing commands, including copying, moving and pasting, flood filling and setting the hot spot.

DRAWING COMMANDS

Here is the list of drawing commands accessible through the buttons at the left side of the application's window. Some commands can be aborted by pressing A inside the bitmap window, allowing the user to select different guiding points where applicable.

Clear
This command clears all bits in the bitmap image. The grid squares will be set to the background color. Pressing C inside the bitmap window has the same effect.

Set This command sets all bits in the bitmap image. The grid squares will be set to the foreground color. Pressing S inside the bitmap window has the same effect.

Invert
This command inverts all bits in the bitmap image. The grid squares will be inverted appropriately. Pressing I inside the bitmap window has the same effect.

Mark
This command is used to mark an area of the grid by dragging out a rectangular shape in the highlighting color. Once the area is marked, it can be operated on by a number of commands (see **Up, Down, Left, Right, Rotate, Flip, Cut,** etc.) Only one marked area can be present at any time. If you attempt to mark another area, the old mark will vanish. The same effect can be achieved by pressing **Shift-MouseButton1** and dragging out a rectangle in the grid window. Pressing **Shift-MouseButton2** will mark the entire grid area.

Unmark
This command will cause the marked area to vanish. The same effect can be achieved by pressing **Shift-MouseButton3**.

Copy
This command is used to copy an area of the grid from one location to another. If there is no marked grid area displayed, **Copy** behaves just like **Mark** described above. Once there is a marked grid area displayed in the highlighting color, this command has two alternative behaviors. If you click a mouse button inside the marked area, you will be able to drag the rectangle that represents the marked area to the desired location. After you release the mouse button, the area will be copied. If you click outside the marked area, **Copy** will assume that you wish to mark a different region of the bitmap image, thus it will behave like **Mark** again.

Move
This command is used to move an area of the grid from one location to another. Its behavior resembles the behavior of **Copy** command, except that the marked area will be moved instead of copied.

Flip Horizontally
This command will flip the bitmap image with respect to the horizontal axes. If a marked area of the grid is highlighted, it will operate only inside the marked area. Pressing F inside the bitmap window

has the same effect.

Up This command moves the bitmap image one pixel up. If a marked area of the grid is highlighted, it will operate only inside the marked area. Pressing UpArrow inside the bitmap window has the same effect.

Flip Vertically
This command will flip the bitmap image with respect to the vertical axes. If a marked area of the grid is highlighted, it will operate only inside the marked area. Pressing V inside the bitmap window has the same effect.

Left
This command moves the bitmap image one pixel to the left. If a marked area of the grid is highlighted, it will operate only inside the marked area. Pressing LeftArrow inside the bitmap window has the same effect.

Fold
This command will fold the bitmap image so that the opposite corners become adjacent. This is useful when creating bitmap images for tiling. Pressing F inside the bitmap window has the same effect.

Right
This command moves the bitmap image one pixel to the right. If a marked area of the grid is highlighted, it will operate only inside the marked area. Pressing RightArrow inside the bitmap window has the same effect.

Rotate Left
This command rotates the bitmap image 90 degrees to the left (counter clockwise.) If a marked area of the grid is highlighted, it will operate only inside the marked area. Pressing L inside the bitmap window has the same effect.

Down
This command moves the bitmap image one pixel down. If a marked area of the grid is highlighted, it will operate only inside the marked area. Pressing DownArrow inside the bitmap window has the same effect.

Rotate Right
This command rotates the bitmap image 90 degrees to the right (clockwise.) If a marked area of the grid is highlighted, it will operate only inside the marked area. Pressing R inside the bitmap window has the same effect.

Point
This command will change the grid squares underneath the mouse pointer if a mouse button is being pressed down. If you drag the mouse button continuously, the line may not be continuous, depending on the speed of your system and frequency of mouse motion events.

Curve
This command will change the grid squares underneath the mouse pointer if a mouse button is being pressed down. If you drag the mouse button continuously, it will make sure that the line is continuous. If your system is slow or *bitmap* receives very few mouse motion events, it might behave quite strangely.

Line
This command will change the gird squares in a line between two squares. Once you press a mouse button in the grid window, *bitmap* will highlight the line from the square where the mouse button was initially pressed to the square where the mouse pointer is located. By releasing the mouse button you will cause the change to take effect, and the highlighted line will disappear.

Rectangle
This command will change the gird squares in a rectangle between two squares. Once you press a mouse button in the grid window, *bitmap* will highlight the rectangle from the square where the mouse button was initially pressed to the square where the mouse pointer is located. By releasing the mouse

button you will cause the change to take effect, and the highlighted rectangle will disappear.

Filled Rectangle

This command is identical to **Rectangle**, except at the end the rectangle will be filled rather than outlined.

Circle

This command will change the gird squares in a circle between two squares. Once you press a mouse button in the grid window, *bitmap* will highlight the circle from the square where the mouse button was initially pressed to the square where the mouse pointer is located. By releasing the mouse button you will cause the change to take effect, and the highlighted circle will disappear.

Filled Circle

This command is identical to **Circle**, except at the end the circle will be filled rather than outlined.

Flood Fill

This command will flood fill the connected area underneath the mouse pointer when you click on the desired square. Diagonally adjacent squares are not considered to be connected.

Set Hot Spot

This command designates one square in the grid as the hot spot if this bitmap image is to be used for defining a cursor. Pressing a mouse button in the desired square will cause a diamond shape to be displayed.

Clear Hot Spot

This command removes any designated hot spot from the bitmap image.

Undo

This command will undo the last executed command. It has depth one, that is, pressing **Undo** after **Undo** will undo itself.

FILE MENU

The File menu commands can be accessed by pressing the File button and selecting the appropriate menu entry, or by pressing Ctrl key with another key. These commands deal with files and global bitmap parameters, such as size, basename, filename etc.

New

This command will clear the editing area and prompt for the name of the new file to be edited. It will not load in the new file.

Load

This command is used to load a new bitmap file into the bitmap editor. If the current image has not been saved, user will be asked whether to save or ignore the changes. The editor can edit only one file at a time. If you need interactive editing, run a number of editors and use cut and paste mechanism as described below.

Insert

This command is used to insert a bitmap file into the image being currently edited. After being prompted for the filename, click inside the grid window and drag the outlined rectangle to the location where you want to insert the new file.

Save

This command will save the bitmap image. It will not prompt for the filename unless it is said to be <none>. If you leave the filename undesignated or –, the output will be piped to stdout.

Save As

This command will save the bitmap image after prompting for a new filename. It should be used if you want to change the filename.

Resize

This command is used to resize the editing area to the new number of pixels. The size should be entered in the WIDTHxHEIGHT format. The information in the image being edited will not be lost unless the new size is smaller that the current image size. The editor was not designed to edit huge files.

Rescale

This command is used to rescale the editing area to the new width and height. The size should be entered in the WIDTHxHEIGHT format. It will not do antialiasing and information will be lost if you rescale to the smaller sizes. Feel free to add you own algorithms for better rescaling.

Filename

This command is used to change the filename without changing the basename nor saving the file. If you specify – for a filename, the output will be piped to stdout.

Basename

This command is used to change the basename, if a different one from the specified filename is desired.

Quit

This command will terminate the bitmap application. If the file was not saved, user will be prompted and asked whether to save the image or not. This command is preferred over killing the process.

EDIT MENU

The Edit menu commands can be accessed by pressing the Edit button and selecting the appropriate menu entry, or by pressing Meta key with another key. These commands deal with editing facilities such as grid, axes, zooming, cut and paste, etc.

Image

This command will display the image being edited and its inverse in its actual size in a separate window. The window can be moved away to continue with editing. Pressing the left mouse button in the image window will cause it to disappear from the screen.

Grid

This command controls the grid in the editing area. If the grid spacing is below the value specified by gridTolerance resource (8 by default), the grid will be automatically turned off. It can be enforced by explicitly activating this command.

Dashed

This command controls the stipple for drawing the grid lines. The stipple specified by dashes resource can be turned on or off by activating this command.

Axes

This command controls the highlighting of the main axes of the image being edited. The actual lines are not part of the image. They are provided to aid user when constructing symmetrical images, or whenever having the main axes highlighted helps your editing.

Stippled

This command controls the stippling of the highlighted areas of the bitmap image. The stipple specified by stipple resource can be turned on or off by activating this command.

Proportional

This command controls the proportional mode. If the proportional mode is on, width and height of all image squares are forced to be equal, regardless of the proportions of the bitmap window.

Zoom

This command controls the zoom mode. If there is a marked area of the image already displayed, bitmap will automatically zoom into it. Otherwise, user will have to highlight an area to be edited in the zoom mode and bitmap will automatically switch into it. One can use all the editing commands and other utilities in the zoom mode. When you zoom out, undo command will undo the whole zoom session.

Cut This commands cuts the contents of the highlighted image area into the internal cut and paste buffer.

Copy

This command copies the contents of the highlighted image area into the internal cut and paste buffer.

Paste

This command will check if there are any other bitmap applications with a highlighted image area, or if there is something in the internal cut and paste buffer and copy it to the image. To place the copied image, click in the editing window and drag the outlined image to the position where you want to place i, and then release the button.

CUT AND PASTE

Bitmap supports two cut and paste mechanisms; the internal cut and paste and the global X selection cut and paste. The internal cut and paste is used when executing copy and move drawing commands and also cut and copy commands from the edit menu. The global X selection cut and paste is used whenever there is a highlighted area of a bitmap image displayed anywhere on the screen. To copy a part of image from another bitmap editor simply highlight the desired area by using the Mark command or pressing the shift key and dragging the area with the left mouse button. When the selected area becomes highlighted, any other applications (such as xterm, etc.) that use primary selection will discard their selection values and unhighlight the appropriate information. Now, use the Paste command for the Edit menu or control mouse button to copy the selected part of image into another (or the same) bitmap application. If you attempt to do this without a visible highlighted image area, the bitmap will fall back to the internal cut and paste buffer and paste whatever was there stored at the moment.

WIDGETS

Below is the widget structure of the *bitmap* application. Indentation indicates hierarchical structure. The widget class name is given first, followed by the widget instance name. All widgets except the bitmap widget are from the standard Athena widget set.

```
Bitmap bitmap
        TransientShell image
                Box box
                        Label normalImage
                        Label invertedImage
        TransientShell input
                Dialog dialog
                        Command okay
                        Command cancel
        TransientShell error
                Dialog dialog
                        Command abort
                        Command retry
        TransientShell qsave
                Dialog dialog
                        Command yes
                        Command no
                        Command cancel
        Paned parent
                Form formy
                        MenuButton fileButton
                        SimpleMenu fileMenu
                                SmeBSB  new
                                SmeBSB  load
                                SmeBSB  insert
                                SmeBSB  save
                                SmeBSB  saveAs
                                SmeBSB  resize
                                SmeBSB  rescale
```

```
                SmeBSB  filename
                SmeBSB  basename
                SmeLine line
                SmeBSB  quit
            MenuButton editButton
            SimpleMenu editMenu
                SmeBSB  image
                SmeBSB  grid
                SmeBSB  dashed
                SmeBSB  axes
                SmeBSB  stippled
                SmeBSB  proportional
                SmeBSB  zoom
                SmeLine line
                SmeBSB  cut
                SmeBSB  copy
                SmeBSB  paste
            Label status
        Pane pane
            Bitmap bitmap
            Form form
                Command clear
                Command set
                Command invert
                Toggle  mark
                Command unmark
                Toggle  copy
                Toggle  move
                Command flipHoriz
                Command up
                Command flipVert
                Command left
                Command fold
                Command right
                Command rotateLeft
                Command down
                Command rotateRight
                Toggle  point
                Toggle  curve
                Toggle  line
                Toggle  rectangle
                Toggle  filledRectangle
                Toggle  circle
                Toggle  filledCircle
                Toggle  floodFill
                Toggle  setHotSpot
                Command clearHotSpot
                Command undo
```

COLORS

If you would like bitmap to be viewable in color, include the following in the #ifdef COLOR section of the file you read with xrdb:

```
*customization:        –color
```

This will cause bitmap to pick up the colors in the app-defaults color customization file: /usr/lib/X11/app-defaults/Bitmap-color.

BITMAP WIDGET

Bitmap widget is a stand-alone widget for editing raster images. It is not designed to edit large images, although it may be used in that purpose as well. It can be freely incorporated with other applications and used as a standard editing tool. The following are the resources provided by the bitmap widget.

Bitmap Widget

Header file	Bitmap.h
Class	bitmapWidgetClass
Class Name	Bitmap
Superclass	Bitmap

All the Simple Widget resources plus . . .

Name	Class	Type	Default Value
foreground	Foreground	Pixel	XtDefaultForeground
highlight	Highlight	Pixel	XtDefaultForeground
framing	Framing	Pixel	XtDefaultForeground
gridTolerance	GridTolerance	Dimension	8
size	Size	String	32x32
dashed	Dashed	Boolean	True
grid	Grid	Boolean	True
stippled	Stippled	Boolean	True
proportional	Proportional	Boolean	True
axes	Axes	Boolean	False
squareWidth	SquareWidth	Dimension	16
squareHeight	SquareHeight	Dimension	16
margin	Margin	Dimension	16
xHot	XHot	Position	NotSet (–1)
yHot	YHot	Position	NotSet (–1)
button1Function	Button1Function	DrawingFunction	Set
button2Function	Button2Function	DrawingFunction	Invert
button3Function	Button3Function	DrawingFunction	Clear
button4Function	Button4Function	DrawingFunction	Invert
button5Function	Button5Function	DrawingFunction	Invert
filename	Filename	String	None ("")
basename	Basename	String	None ("")

NAME

editres - a dynamic resource editor for X Toolkit applications

SYNTAX

editres [*-toolkitoption* ...]

OPTIONS

Editres accepts all of the standard X Toolkit command line options (see *X(1)*). The order of the command line options is not important.

DESCRIPTION

Editres is a tool that allows users and application developers to view the full widget hierarchy of any X Toolkit client that speaks the Editres protocol. In addition editres will help the user construct resource specifications, allow the user to apply the resource to the application and view the results dynamically. Once the user is happy with a resource specification editres will append the resource string to the user's X Resources file.

USING EDITRES

Editres provides a window consisting of the following four areas:

Menu Bar	A set of popup menus that allow you full access to editres's features.
Panner	The panner allows a more intuitive way to scroll the application tree display.
Message Area	Displays information to the user about the action that editres expects of her.
Application Widget Tree	This area will be used to display the selected client's widget tree.

To begin an editres session select the **Get Widget Tree** menu item from the command menu. This will change the pointer cursor to cross hair. You should now select the application you wish look at by clicking on any of its windows. If this application understands the editres protocol then editres will display the client's widget tree in its tree window. If the application does not understand the editres protocol editres will inform you of this fact in the message area after a few seconds delay.

Once you have a widget tree you may now select any of the other menu options. The effect of each of these is described below.

COMMANDS

Send Widget Tree

Allows the user to click on any client that speaks the editres protocol and receive its widget tree.

Refresh Widget Tree

Editres only knows about the widgets that exist at the present time. Many applications create and destroy widgets "on-the-fly". Selecting this menu item will cause editres to ask the application to resend its widget tree, thus updating its information to the new state of the application.

Example Xman only creates the widgets for its *topbox* when it starts up. None of the widgets for the manual page window are created until the user actually clicks on the *Manual Page* button. If you retrieved xman's widget tree before the the manual page is active, you may wish to refresh the widget tree after the manual page has been displayed. This will allow you to also edit the manual page's resources.

Dump Widget Tree to a File

For documenting applications it is often useful to be able to dump the entire application widget tree to an ascii file. This file can then be included in the manual page. When this menu item is selected a popup dialog is activated. Type the name of the file in this dialog, and either select *okay*, or type a carriage-return. Editres will now dump the widget tree to this file. To cancel the file dialog just select the *cancel* button.

Show Resource Box

This command will popup a resource box for the current client. This resource box (described in detail below) will allow the user to see exactly which resources can be set for the widget that is currently selected in the widget tree display. Only one widget may be currently selected, if greater or fewer are selected editres will refuse to pop up the resource box, and put an error message in the **Message Area**.

Set Resource
: This command will popup a simple dialog box for setting an arbitrary resource on all selected widgets. You must type in the resource name, as well as the value. You can use the Tab key to switch between the resource name field the resource value field.

Quit
: Exits editres.

TREE COMMANDS

The **Tree** menu contains several commands that allow operations to be performed on the widget tree.

Select Widget in Client
: This menu item allows you to select any widget in the application, editres will then highlight the corrosponding element the widget tree display. Once this menu item is selected the pointer cursor will again turn to a crosshair, and you must click any pointer button in the widget you wish to have displayed. Since some widgets are fully obscured by their children, it is not possible to get to every widget this way, but this mechanism does give very useful feedback between the elements in the widget tree and those in the actual client.

Select All
Unselect All
Invert All
: These functions allow the user to select, unselect, or invert all widgets in the widget tree.

Select Children
Select Parents
: These functions select the immediate parent or children of each of the currently selected widgets.

Select Descendants
Select Ancestors
: These functions select all parents or children of each of the currently selected widgets. This is a recursive search.

Show Widget Names
Show Class Names
Show Widget Windows
: When the tree widget is initially displayed the labels of each widget in the tree correspond to the widget names. These functions will cause the label of **all** widgets in the tree to be changed to show the class name, IDs, or window associated with each widget in the application. The widget IDs, and windows are shown as hex numbers.

In addition there are keyboard accelerators for each of the Tree operations. If the input focus is over an individual widget in the tree, then that operation will only effect that widget. If the input focus is in the Tree background it will have exactly the same effect as the corresponding menu item.

The translation entries shown may be applied to any widget in the application. If that widget is a child of the Tree widget, then it will only affect that widget, otherwise it will have the same effect as the commands in the tree menu.

Flash Active Widgets
: This command is the inverse of the **Select Widget in Client** command, it will show the user each widget that is currently selected in the widget tree, by flashing the corresponding widget in the application *numFlashes* (three by default) times in the *flashColor*.

Key	**Option**	**Translation Entry**
space	Unselect	Select(nothing)

w	Select	Select(widget)
s	Select	Select(all)
i	Invert	Select(invert)
c	Select Children	Select(children)
d	Select Descendants	Select(descendants)
p	Select Parent	Select(parent)
a	Select Ancestors	Select(ancestors)
N	Show Widget Names	Relabel(name)
C	Show Class Names	Relabel(class)
I	Show Widget IDs	Relabel(id)
W	Show Widget Windows	Relabel(window)
T	Toggle Widget/Class Name	Relabel(toggle)

Clicking button 1 on a widget adds it to the set of selected widgets. Clicking button 2 on a widget deselects all other widgets and then selects just that widget. Clicking button 3 on a widget toggles its label between the widget's instance name the widget's class name.

USING THE RESOURCE BOX

The resource box contains five different areas. Each of the areas, as they appear on the screen, from top to bottom will be discussed.

The Resource Line

This area at the top of the resource box shows the current resource name exactly as it would appear if you were to save it to a file or apply it.

The Widget Names and Classes

This area allows you to select exactly which widgets this resource will apply to. The area contains four lines, the first contains the name of the selected widget and all its ancestors, and the more restrictive dot (**.**) separator. The second line contains less specific the Class names of each widget, and well as the less restrictive star (*) separator. The third line contains a set of special buttons called **Any Widget** which will generalize this level to match any widget. The last line contains a set of special buttons called **Any Widget Chain** which will turn the single level into something that matches zero or more levels.

The initial state of this area is the most restrictive, using the resource names and the dot separator. By selecting the other buttons in this are you can ease the restrictions to allow more and more widgets to match the specification. The extreme case is to select all the **Any Widget Chain** buttons, which will match every widget in the application. As you select different buttons the tree display will update to show you exactly which widgets will be effected by the current resource specification.

Normal and Constraint Resources

The next area allows you to select the name of the normal or constraint resources you wish to set. Some widgets may not have constraint resources, so that area will not appear.

Resource Value

This next area allows you to enter the resource value. This value should be entered exactly as you would type a line into your resource file. Thus it should contain no unescaped new-lines. There are a few special character sequences for this file:

\n - This will be replaced with a newline.

\### - Where # is any octal digit. This will be 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.

\<new-line> - This will compress to nothing.

\\ - This will compress to a single backslash.

Command Area
This area contains several command buttons that I will describe in this section.

Set Save File
This button allows the user to modify file that the resources will be saved to. This button will bring up a dialog box that will ask you for a filename, once the filename has been entered, either hit carriage-return or click on the *okay* button. To popdown the dialog box without changing the save file, click the *cancel* button.

Save
This button will append the **resource line** described above to the end of the current save file. If no save file has been set the **Set Save File** dialog box will be popped up to prompt the user for a filename.

Apply
This button attempts to perform a XtSetValues call on all widgets that match the **resource line** described above. The value specified is applied directly to all matching widgets. This behavior is an attempt to give a dynamic feel to the resource editor. Since this feature allows users to put an application in states it may not be willing to handle, a hook has been provided to allow specific clients block these SetValues requests (see **Blocking Editres Requests** below).

Unfortunately due to design constraints imposed on the widgets by the X Toolkit and the Resource Manager, trying to coerce an inherently static system into dynamic behavior can cause strange results. There is no guarantee that the results of an apply will be the same as what will happen when you save the value, and restart the application. This functionality is provided to try to give you a rough feel for what your changes will accomplish, and the results obtained should be considered suspect at best. Having said that, this is one of the neatest features of editres, and I strongly suggest that you play with it, and see what it can do.

Save and Apply
This button combines the Save and Apply actions described above into one button.

Popdown Resource Box
This button will remove the resource box from the display.

BLOCKING EDITRES REQUESTS

The editres protocol has been built into the Athena Widget set. This allows all application that are linked against Xaw to be able to speak to the resource editor. While this provides great flexability, and is a useful tool, it can quite easily be abused. It is therefore possible for any Xaw client to specify a value for the **editresBlock** resource described below, to keep editres from divulging information about its internals, or to disable the **SetValues** part of the protocol.

editresBlock (Class **EditresBlock)**
Specifies which type of blocking this client wishes to impose on the editres protocol.

The accepted values are:

all
Block all requests.

setValues
Block all setvalues request, this is the only editres request that actually modifies the application, this is in effect stating that the applicaion is read-only.

none
Allow all editres requests.

Remember that these resources are set on any Xaw client, **not editres**. They allow individual clients to keep all or some of the requests editres makes from ever succeeding. Of course, editres is also an Xaw client, so it may also be viewed and modified by editres (rather recursive, I know), these commands can be blocked by setting the **editresBlock** resource on editres itself.

RESOURCES

For *editres* the available application resources are:

numFlashes (Class **NumFlashes)**

Specifies the number of times the widgets in the client application will be flashed when the **Show Active Widgets** command in invoked.

flashTime (Class **FlashTime)**
Amount of time between the flashes described above.

flashColor (Class **flashColor)**
Specifies the color used to flash client widgets. A bright color should be used that will immediately draw your attention to the area being flashed, such as red or yellow.

saveResourcesFile (Class **SaveResourcesFile)**
This is the file the resource line will be append to when the **Save** button activated in the resource box.

WIDGETS

In order to specify resources, it is useful to know the hierarchy of the widgets which compose *editres*. In the notation below, indentation indicates hierarchical structure. The widget class name is given first, followed by the widget instance name.

```
Editres  editres
        Paned  paned
                Box  box
                        MenuButton  commands
                                SimpleMenu  menu
                                SmeBSB  sendTree
                                SmeBSB  refreshTree
                                SmeBSB  dumpTreeToFile
                                SmeLine  line
                                SmeBSB  getResourceList
                                SmeLine  line
                                SmeBSB  quit
                        MenuButton  treeCommands
                                SimpleMenu  menu
                                SmeBSB  showClientWidget
                                SmeBSB  selectAll
                                SmeBSB  unselectAll
                                SmeBSB  invertAll
                                SmeLine  line
                                SmeBSB  selectChildren
                                SmeBSB  selectParent
                                SmeBSB  selectDescendants
                                SmeBSB  selectAncestors
                                SmeLine  line
                                SmeBSB  showWidgetNames
                                SmeBSB  showClassNames
                                SmeBSB  showWidgetIDs
                                SmeBSB  showWidgetWindows
                                SmeLine  line
                                SmeBSB  flashActiveWidgets
                Paned  hPane
                        Panner  panner
                        Label  userMessage
                        Grip  grip
                Porthole  porthole
                        Tree  tree
                                Toggle  <name of widget in client>
```

```
                    .
                    .
                    .
                    TransientShell  resourceBox
                    Paned  pane
                    Label  resourceLabel
                    Form  namesAndClasses
                    Toggle  dot
                    Toggle  star
                    Toggle  any
                    Toggle  name
                    Toggle  class
                        .
                        .
                        .
                    Label  namesLabel
                    List  namesList
                    Label  constraintLabel
                    List  constraintList
                    Form  valueForm
                    Label  valueLabel
                    Text  valueText
                    Box  commandBox
                    Command  setFile
                    Command  save
                    Command  apply
                    Command  saveAndApply
                    Command  cancel
                    Grip  grip
            Grip  grip
```

ENVIRONMENT

DISPLAY

to get the default host and display number.

XENVIRONMENT

to get the name of a resource file that overrides the global resources stored in the RESOURCE_MANAGER property.

FILES

/usr/lib/X11/app-defaults/Editres - specifies required resources

SEE ALSO

X(1), xrdb(1), Athena Widget Set

RESTRICTIONS

This is a prototype, there are lots of nifty features I would love to add, but I hope this will give you some ideas about what a resource editor can do.

COPYRIGHT

AUTHOR

Chris D. Peterson, formerly MIT X Consortium

NAME

fs – X font server

SYNOPSIS

fs [-config *configuration_file*] [-port *tcp_port*]

DESCRIPTION

Fs is the X Window System font server. It supplies fonts to X Window System display servers.

STARTING THE SERVER

The server is usually run by a system administrator, and started via boot files like */etc/rc.local*. Users may also wish to start private font servers for specific sets of fonts.

OPTIONS

–config configuration_file
Specifies the configuration file the font server will use.

–ls listen-socket
Specifies a file descriptor which is already set up to be used as the listen socket. This option is only intended to be used by the font server itself when automatically spawning another copy of itself to handle additional connections.

–port tcp_port
Specifies the TCP port number on which the server will listen for connections.

SIGNALS

SIGTERM
This causes the font server to exit cleanly.

SIGUSR1
This signal is used to cause the server to re-read its configuration file.

SIGUSR2
This signal is used to cause the server to flush any cached data it may have.

SIGHUP This signal is used to cause the server to reset, closing all active connections and re-reading the configuration file.

CONFIGURATION

The configuration language is a list of keyword and value pairs. Each keyword is followed by an '=' and then the desired value.

Recognized keywords include:

in the first column, a comment character

catalogue (list of string)
Ordered list of font path element names.

alternate-servers (list of string)
List of alternate servers for this font server.

client-limit (cardinal)
Number of clients this font server will support before refusing service. This is useful for tuning the load on each individual font server.

clone-self (boolean)
Whether this font server should attempt to clone itself when it reachs the client-limit.

default-point-size (cardinal)
The default pointsize (in decipoints) for fonts that don’t specify.

default-resolutions (list of resolutions)
Resolutions the server supports by default. This information may be used as a hint for pre-rendering, and substituted for scaled fonts which do not specify a resolution.

error-file (string)
Filename of the error file. All warnings and errors will be logged here.

port (cardinal)
TCP port on which the server will listen for connections.

use-syslog (boolean)
Whether syslog(3) (on supported systems) is to be used for errors.

EXAMPLE

```
#
# sample font server configuration file
#

# allow a max of 10 clients to connect to this font server
client-limit = 10

# when a font server reaches its limit, start up a new one
clone-self = on

# alternate font servers for clients to use
alternate-servers = hansen:7001,hansen:7002

# where to look for fonts
# the first is a set of Speedo outlines, the second is a set of
# misc bitmaps and the last is a set of 100dpi bitmaps
#
catalogue = /usr/lib/fonts/speedo,
        /usr/lib/X11/ncd/fonts/misc,
        /usr/lib/X11/ncd/fonts/100dpi/

# in 12 points, decipoints
default-point-size = 120

# 100 x 100 and 75 x 75
default-resolutions = 100,100,75,75
```

SEE ALSO

X(1), *Font server implementation overview*

COPYRIGHT

AUTHORS

Dave Lemke, Network Computing Devices, Inc
Keith Packard, Massachusetts Institute of Technology

NAME

fsinfo – font server information utility

SYNOPSIS

fsinfo [–server *servername*]

DESCRIPTION

Fsinfo is a utility for displaying information about an X font server. It is used to examine the capabilities of a server, the predefined values for various parameters used in communicating between clients and the server, and the font catalogues and alternate servers that are available.

EXAMPLE

The following shows a sample produced by *fsinfo.*

```
name of server:    hansen:7000
version number:    1
vendor string:     Font Server Prototype
vendor release number:     17
maximum request size:      16384 longwords (65536 bytes)
number of catalogues:      1
        all
Number of alternate servers: 2
    #0      hansen:7001
    #1      hansen:7002
number of extensions:      0
```

ENVIRONMENT

FONTSERVER

To get the default fontserver.

SEE ALSO

fs(1), fslsfonts(1)

COPYRIGHT

AUTHOR

Dave Lemke, Network Computing Devices, Inc

NAME

fslsfonts - server font list displayer for X font server

SYNOPSIS

fslsfonts [-options ...] [-fn pattern]

DESCRIPTION

Fslsfonts lists the fonts that match the given *pattern*. The wildcard character "*" may be used to match any sequence of characters (including none), and "?" to match any single character. If no pattern is given, "*" is assumed.

The "*" and "?" characters must be quoted to prevent them from being expanded by the shell.

OPTIONS

–server *host*:*port*
: This option specifies the X font server to contact.

–l[l[l]] This option indicates that medium, long, and very long listings, respectively, should be generated for each font.

–m This option indicates that long listings should also print the minimum and maximum bounds of each font.

–C This option indicates that listings should use multiple columns. This is the same as **-n 0**.

–1 This option indicates that listings should use a single column. This is the same as **-n 1**.

–w *width*
: This option specifies the width in characters that should be used in figuring out how many columns to print. The default is 79.

–n *columns*
: This option specifies the number of columns to use in displaying the output. By default, it will attempt to fit as many columns of font names into the number of character specified by **-w** *width*.

–u This option indicates that the output should be left unsorted.

SEE ALSO

fs(1), showfont(1), xlsfonts(1)

ENVIRONMENT

FONTSERVER
: to get the default host and port to use.

BUGS

Doing ‘‘fslsfonts -l’’ can tie up your server for a very long time. This is really a bug with single-threaded non-preemptable servers, not with this program.

COPYRIGHT

AUTHOR

Dave Lemke, Network Computing Devices, Inc

NAME

fstobdf - BDF font generator

SYNOPSIS

fstobdf -fn *fontname* [-option ...]

DESCRIPTION

The *fstobdf* program reads a font from a font server and prints a BDF file on the standard output that may be used to recreate the font. This is useful in testing servers, debugging font metrics, and reproducing lost BDF files.

OPTIONS

–server *servername*

This option specifies the server from which the font should be read.

–fn *fontname*

This option specifies the font for which a BDF file should be generated.

SEE ALSO

fs(1), bdftosnf(1), fslsfonts(1), xtobdf(1)

COPYRIGHT

AUTHOR

Olaf Brandt, Network Computing Devices
Dave Lemke, Network Computing Devices

Jim Fulton, MIT X Consortium

NAME

ico – animate an icosahedron or other polyhedron

SYNOPSIS

ico [-display display] [-geometry geometry] [-r] [-d pattern] [-i] [-dbl] [-faces] [-noedges] [-sleep n] [-obj object] [-objhelp] [-colors color-list]

DESCRIPTION

Ico displays a wire-frame rotating polyhedron, with hidden lines removed, or a solid-fill polyhedron with hidden faces removed. There are a number of different polyhedra available; adding a new polyhedron to the program is quite simple.

OPTIONS

-r Display on the root window instead of creating a new window.

-d pattern
Specify a bit pattern for drawing dashed lines for wire frames.

-i Use inverted colors for wire frames.

-dbl Use double buffering on the display. This works for either wire frame or solid fill drawings. For solid fill drawings, using this switch results in substantially smoother movement. Note that this requires twice as many bit planes as without double buffering. Since some colors are typically allocated by other programs, most eight-bit-plane displays will probably be limited to eight colors when using double buffering.

-faces Draw filled faces instead of wire frames.

-noedges
Don't draw the wire frames. Typically used only when -faces is used.

-sleep *n* Sleep n seconds between each move of the object.

-obj *object*
Specify what object to draw. If no object is specified, an icosahedron is drawn.

-objhelp
Print out a list of the available objects, along with information about each object.

-colors *color color ...*
Specify what colors should be used to draw the filled faces of the object. If less colors than faces are given, the colors are reused.

ADDING POLYHEDRA

If you have the source to ico, it is very easy to add more polyhedra. Each polyhedron is defined in an include file by the name of objXXX.h, where XXX is something related to the name of the polyhedron. The format of the include file is defined in the file polyinfo.h. Look at the file objcube.h to see what the exact format of an objXXX.h file should be, then create your objXXX.h file in that format.

After making the new objXXX.h file (or copying in a new one from elsewhere), simply do a 'make depend'. This will recreate the file allobjs.h, which lists all of the objXXX.h files. Doing a 'make' after this will rebuild ico with the new object information.

SEE ALSO

X(1)

BUGS

Pyramids and tetrahedrons with filled faces do not display correctly.

A separate color cell is allocated for each name in the -colors list, even when the same name may be specified twice.

COPYRIGHT

NAME

imake – C preprocessor interface to the make utility

SYNOPSIS

imake [**–D***define*] [**–I***dir*] [**–T***template*] [**–f** *filename*] [**–s** *filename*] [**–e**] [**–v**]

DESCRIPTION

Imake is used to generate *Makefiles* from a template, a set of *cpp* macro functions, and a per-directory input file called an *Imakefile*. This allows machine dependencies (such has compiler options, alternate command names, and special *make* rules) to be kept separate from the descriptions of the various items to be built.

OPTIONS

The following command line options may be passed to *imake*:

–D*define* This option is passed directly to *cpp*. It is typically used to set directory-specific variables. For example, the X Window System uses this flag to set *TOPDIR* to the name of the directory containing the top of the core distribution and *CURDIR* to the name of the current directory, relative to the top.

–I*directory*
This option is passed directly to *cpp*. It is typically used to indicate the directory in which the *imake* template and configuration files may be found.

–T*template*
This option specifies the name of the master template file (which is usually located in the directory specified with *–I*) used by *cpp*. The default is *Imake.tmpl*.

–f *filename*
This option specifies the name of the per-directory input file. The default is *Imakefile*.

–s *filename*
This option specifies the name of the *make* description file to be generated but *make* should not be invoked. If the *filename* is a dash (–), the output is written to *stdout*. The default is to generate, but not execute, a *Makefile*.

–e This option indicates the *imake* should execute the generated *Makefile*. The default is to leave this to the user.

–v This option indicates that *imake* should print the *cpp* command line that it is using to generate the *Makefile*.

HOW IT WORKS

Imake invokes *cpp* with any *–I* or *–D* flags passed on the command line and passes it the following 3 lines:

```
#define IMAKE_TEMPLATE "Imake.tmpl"
#define INCLUDE_IMAKEFILE "Imakefile"
#include IMAKE_TEMPLATE
```

where *Imake.tmpl* and *Imakefile* may be overridden by the *–T* and *–f* command options, respectively.

The IMAKE_TEMPLATE typically reads in a file containing machine-dependent parameters (specified as *cpp* symbols), a site-specific parameters file, a file defining variables, a file containing *cpp* macro functions for generating *make* rules, and finally the *Imakefile* (specified by INCLUDE_IMAKEFILE) in the current directory. The *Imakefile* uses the macro functions to indicate what targets should be built; *imake* takes care of generating the appropriate rules.

Imake configuration files contain two types of variables, imake variables and make variables. The imake variables are interpreted by cpp when *imake* is run. By convention they are mixed case. The make variables are written into the *Makefile* for later interpretation by *make*. By convention make variables are upper case.

The rules file (usually named *Imake.rules* in the configuration directory) contains a variety of *cpp* macro functions that are configured according to the current platform. *Imake* replaces any occurrences of the string ‘‘@@’’ with a newline to allow macros that generate more than one line of *make* rules. For example, the macro

```
#define      program_target(program, objlist)              @@\
program:     objlist                                       @@\
             $(CC)  –o  $@  objlist  $(LDFLAGS)
```

when called with *program_target(foo, foo1.o foo2.o)* will expand to

```
foo:         foo1.o  foo2.o
             $(CC)  –o  $@  foo1.o  foo2.o  $(LDFLAGS)
```

On systems whose *cpp* reduces multiple tabs and spaces to a single space, *imake* attempts to put back any necessary tabs (*make* is very picky about the difference between tabs and spaces). For this reason, colons (:) in command lines must be preceded by a backslash (\).

USE WITH THE X WINDOW SYSTEM

The X Window System uses *imake* extensively, for both full builds within the source tree and external software. As mentioned above, two special variables, *TOPDIR* and *CURDIR,* are set to make referencing files using relative path names easier. For example, the following command is generated automatically to build the *Makefile* in the directory *lib/X/* (relative to the top of the sources):

```
% ../.././config/imake  –I../.././config  \
        –DTOPDIR=../../.   –DCURDIR=./lib/X
```

When building X programs outside the source tree, a special symbol *UseInstalled* is defined and *TOPDIR* and *CURDIR* are omitted. If the configuration files have been properly installed, the script *xmkmf*(1) may be used.

INPUT FILES

Here is a summary of the files read by *imake* as used by X. The indentation shows what files include what other files.

```
Imake.tmpl                    generic variables
   site.def                   site-specific, BeforeVendorCF defined
   ∗.cf                       machine-specific
      ∗Lib.rules              shared library rules
   site.def                   site-specific, AfterVendorCF defined
   Project.tmpl               X-specific variables
      ∗Lib.tmpl               shared library variables
   Imake.rules                rules
Imakefile
   Library.tmpl               library rules
   Server.tmpl                server rules
```

Note that *site.def* gets included twice, once before the ∗*.cf* file and once after. Although most site customizations should be specified after the ∗*.cf* file, some, such as the choice of compiler, need to be specified before, because other variable settings may depend on them.

The first time *site.def* is included, the variable BeforeVendorCF is defined, and the second time, the variable AfterVendorCF is defined. All code in *site.def* should be inside an #ifdef for one of these symbols.

FILES

/usr/tmp/tmp-imake.*nnnnnn*	temporary input file for cpp
/usr/tmp/tmp-make.*nnnnnn*	temporary input file for make
/lib/cpp	default C preprocessor

SEE ALSO

make(1), xmkmf(1)
S. I. Feldman, *Make — A Program for Maintaining Computer Programs*

ENVIRONMENT VARIABLES

The following environment variables may be set, however their use is not recommended as they introduce dependencies that are not readily apparent when *imake* is run:

IMAKEINCLUDE
If defined, this should be a valid include argument for the C preprocessor. E.g., ‘‘–I/usr/include/local’’. Actually, any valid *cpp* argument will work here.

IMAKECPP
If defined, this should be a valid path to a preprocessor program. E.g. ‘‘/usr/local/cpp’’. By default, *imake* will use /lib/cpp.

IMAKEMAKE
If defined, this should be a valid path to a make program, such as ‘‘/usr/local/make’’. By default, *imake* will use whatever *make* program is found using *execvp(3).* This variable is only used if the ‘‘–e’’ option is specified.

AUTHOR

Todd Brunhoff, Tektronix and MIT Project Athena; Jim Fulton, MIT X Consortium

NAME

listres - list resources in widgets

SYNOPSIS

listres [-option ...]

DESCRIPTION

The *listres* program generates a list of a widget's resource database. The class in which each resource is first defined, the instance and class name, and the type of each resource is listed. If no specific widgets or the *-all* switch are given, a two-column list of widget names and their class hierarchies is printed.

OPTIONS

Listres accepts all of the standard toolkit command line options along with those listed below:

–all This option indicates that *listres* should print information for all known widgets and objects.

–nosuper
This option indicates that resources that are inherited from a superclass should not be listed. This is useful for determining which resources are new to a subclass.

–variable
This option indicates that widgets should be identified by the names of the class record variables rather than the class name given in the variable. This is useful for distinguishing subclasses that have the same class name as their superclasses.

–top *name*
This option specifies the name of the widget to be treated as the top of the hierarchy. Case is not significant, and the name may match either the class variable name or the class name. The default is ‘‘core’’.

–format *printf–string*
This option specifies the *printf*-style format string to be used to print out the name, instance, class, and type of each resource.

X DEFAULTS

To be written.

SEE ALSO

X(1), xrdb(1), appropriate widget documents

BUGS

On operating systems that do not support dynamic linking of run-time routines, this program must have all of its known widgets compiled in. The sources provide several tools for automating this process for various widget sets.

COPYRIGHT

AUTHOR

Jim Fulton, MIT X Consortium

NAME

lndir – create a shadow directory of symbolic links to another directory tree

SYNOPSIS

lndir fromdir [todir]

DESCRIPTION

Lndir makes a shadow copy *todir* of a directory tree *fromdir*, except that the shadow is not populated with real files but instead with symbolic links pointing at the real files in the *fromdir* directory tree. This is usually useful for maintaining source code for different machine architectures. You create a shadow directory containing links to the real source which you will have usually NFS mounted from a machine of a different architecture, and then recompile it. The object files will be in the shadow directory, while the source files in the shadow directory are just symlinks to the real files.

This has the advantage that if you update the source, you need not propagate the change to the other architectures by hand, since all source in shadow directories are symlinks to the real thing: just cd to the shadow directory and recompile away.

The *todir* argument is optional and defaults to the current directory. The *fromdir* argument may be relative (e.g., ../src) and is relative to *todir* (not the current directory).

Note that RCS and SCCS directories are not shadowed.

Note that if you add files, you must run *lndir* again. Deleting files is a more painful problem; the symlinks will just point into never never land.

BUGS

Patch gets upset if it cannot change the files. You should never run *patch* from a shadow directory anyway.

You need to use something like

```
find todir -type l -print | xargs rm
```

to clear out all files before you can relink (if fromdir moved, for instance). Something like

```
find . \! -type d -print
```

will find all files that are not directories.

NAME

makedepend – create dependencies in makefiles

SYNOPSIS

makedepend [**–Dname=def**] [**–Dname**] [**–Iincludedir**] [**–a**] [**–fmakefile**] [**–oobjsuffix**] [**–pobjprefix**] [**–sstring**] [**–wwidth**] [**– – otheroptions – –**] sourcefile ...

DESCRIPTION

Makedepend reads each *sourcefile* in sequence and parses it like a C-preprocessor, processing all *#include, #define, #undef, #ifdef, #ifndef, #endif, #if* and *#else* directives so that it can correctly tell which *#include,* directives would be used in a compilation. Any *#include,* directives can reference files having other *#include* directives, and parsing will occur in these files as well.

Every file that a *sourcefile* includes, directly or indirectly, is what **makedepend** calls a "dependency". These dependencies are then written to a *makefile* in such a way that **make(1)** will know which object files must be recompiled when a dependency has changed.

By default, **makedepend** places its output in the file named *makefile* if it exists, otherwise *Makefile.* An alternate makefile may be specified with the *-f* option. It first searches the makefile for the line

```
# DO NOT DELETE THIS LINE -- make depend depends on it.
```

or one provided with the *-s* option, as a delimiter for the dependency output. If it finds it, it will delete everything following this to the end of the makefile and put the output after this line. If it doesn't find it, the program will append the string to the end of the makefile and place the output following that. For each *sourcefile* appearing on the command line, **makedepend** puts lines in the makefile of the form

```
sourcefile.o: dfile ...
```

Where "sourcefile.o" is the name from the command line with its suffix replaced with ".o", and "dfile" is a dependency discovered in a *#include* directive while parsing *sourcefile* or one of the files it included.

EXAMPLE

Normally, **makedepend** will be used in a makefile target so that typing "make depend" will bring the dependencies up to date for the makefile. For example,

```
SRCS = file1.c file2.c ...
CFLAGS = -O -DHACK -I../foobar -xyz
depend:
        makedepend -- $(CFLAGS) -- $(SRCS)
```

OPTIONS

Makedepend will ignore any option that it does not understand so that you may use the same arguments that you would for **cc(1).**

–Dname=def or –Dname
Define. This places a definition for *name* in **makedepend's** symbol table. Without *=def* the symbol becomes defined as "1".

–Iincludedir
Include directory. This option tells **makedepend** to prepend *includedir* to its list of directories to search when it encounters a *#include* directive. By default, **makedepend** only searches /usr/include.

–a Append the dependencies to the end of the file instead of replacing them.

–fmakefile
Filename. This allows you to specify an alternate makefile in which **makedepend** can place its output.

–oobjsuffix
Object file suffix. Some systems may have object files whose suffix is something other than ".o".

This option allows you to specify another suffix, such as ".b" with *-o.b* or ":obj" with *-o:obj* and so forth.

–pobjprefix
Object file prefix. The prefix is prepended to the name of the object file. This is usually used to designate a different directory for the object file. The default is the empty string.

–sstring
Starting string delimiter. This option permits you to specify a different string for **makedepend** to look for in the makefile.

–wwidth
Line width. Normally, **makedepend** will ensure that every output line that it writes will be no wider than 78 characters for the sake of readability. This option enables you to change this width.

– – options – –
If **makedepend** encounters a double hyphen (– –) in the argument list, then any unrecognized argument following it will be silently ignored; a second double hyphen terminates this special treatment. In this way, **makedepend** can be made to safely ignore esoteric compiler arguments that might normally be found in a CFLAGS **make** macro (see the **EXAMPLE** section above). All options that **makedepend** recognizes and appear between the pair of double hyphens are processed normally.

ALGORITHM

The approach used in this program enables it to run an order of magnitude faster than any other "dependency generator" I have ever seen. Central to this performance are two assumptions: that all files compiled by a single makefile will be compiled with roughly the same *-I* and *-D* options; and that most files in a single directory will include largely the same files.

Given these assumptions, **makedepend** expects to be called once for each makefile, with all source files that are maintained by the makefile appearing on the command line. It parses each source and include file exactly once, maintaining an internal symbol table for each. Thus, the first file on the command line will take an amount of time proportional to the amount of time that a normal C preprocessor takes. But on subsequent files, if it encounter's an include file that it has already parsed, it does not parse it again.

For example, imagine you are compiling two files, *file1.c* and *file2.c,* they each include the header file *header.h,* and the file *header.h* in turn includes the files *def1.h* and *def2.h.* When you run the command

```
makedepend file1.c file2.c
```

makedepend will parse *file1.c* and consequently, *header.h* and then *def1.h* and *def2.h.* It then decides that the dependencies for this file are

```
file1.o: header.h def1.h def2.h
```

But when the program parses *file2.c* and discovers that it, too, includes *header.h,* it does not parse the file, but simply adds *header.h, def1.h* and *def2.h* to the list of dependencies for *file2.o.*

SEE ALSO

cc(1), make(1)

BUGS

If you do not have the source for cpp, the Berkeley C preprocessor, then **makedepend** will be compiled in such a way that all *#if* directives will evaluate to "false" regardless of their actual value. This may cause the wrong *#include* directives to be evaluated. **Makedepend** should simply have its own parser written for *#if* expressions.

Imagine you are parsing two files, say *file1.c* and *file2.c,* each includes the file *def.h.* The list of files that *def.h* includes might truly be different when *def.h* is included by *file1.c* than when it is included by *file2.c.* But once **makedepend** arrives at a list of dependencies for a file, it is cast in concrete.

AUTHOR

Todd Brunhoff, Tektronix, Inc. and MIT Project Athena

NAME

maze – an automated X11 demo repeatedly creating and solving a random maze

SYNTAX

maze [**–S**] [**–r**] [**–g** *geometry*] [**–d** *display*]

DESCRIPTION

The *maze* program creates a "random" maze and then solves it with graphical feedback.

Command Options

–S Full screen window option...

–r Reverse video option...

–g *geometry*
Specifies the window geometry to be used...

–d *display*
Specifies the display to be used...

The following lists the current functionality of various mouse button clicks;

LeftButton
Clears the window and restarts maze...

MiddleButton
Toggles the maze program, first click -> *stop*, second click -> *continue*...

RightButton
Kills maze...

LIMITATIONS

No color support...
Expose events force a restart of maze...
Currently, mouse actions are based on "raw" values [Button1, Button2 and Button3] from the ButtonPress event...
[doesn't use pointer mapping]

COPYRIGHT

AUTHOR(s)

Richard Hess [X11 extensions] {...}!uunet!cimshop!rhess
Consilium, Mountain View, CA
Dave Lemke [X11 version] lemke@sun.COM

Sun MicroSystems, Mountain View, CA
Martin Weiss [SunView version]
Sun MicroSystems, Mountain View, CA

NAME

mkdirhier – makes a directory hierarchy

SYNOPSIS

mkdirhier directory ...

DESCRIPTION

The *mkdirhier* command creates the specified directories. Unlike *mkdir* if any of the parent directories of the specified directory do not exist, it creates them as well.

SEE ALSO

mkdir(1)

NAME

mkfontdir - create fonts.dir file from directory of font files

SYNOPSIS

mkfontdir [directory-names]

DESCRIPTION

Mkfontdir For each directory argument, mkfontdir reads all of the font files in the directory searching for properties named "FONT", or (failing that) the name of the file stripped of its suffix. These are used as font names, which are written out to the file "fonts.dir" in the directory along with the name of the font file.

The kinds of font files read by mkfontdir depends on configuration parameters, but typically include PCF (suffix ".pcf"), SNF (suffix ".snf") and BDF (suffix ".bdf"). If a font exists in multiple formats, *mkfontdir* will first choose PCF, then SNF and finally BDF.

SCALABLE FONTS

Because scalable font files do not usually include the X font name, the fonts.dir file in directories containing such fonts must be edited by hand to include the appropriate entries for those fonts. However, when *mkfontdir* is run, all of those additions will be lost, so be careful.

FONT NAME ALIASES

The file "fonts.alias" which can be put in any directory of the font-path is used to map new names to existing fonts, and should be edited by hand. The format is straight forward enough, two white-space separated columns, the first containing aliases and the second containing font-name patterns.

When a font alias is used, the name it references is search for in the normal manner, looking through each font directory in turn. This means that the aliases need not mention fonts in the same directory as the alias file.

To embed white-space in either name, simply enclose them in double-quote marks, to embed double-quote marks (or any other character), precede them with back-slash:

```
"magic-alias with spaces"  "\"font\name\" with quotes"
regular-alias              fixed
```

If the string "FILE_NAMES_ALIASES" stands alone on a line, each file-name in the directory (stripped of it’s suffix) will be used as an alias for that font.

USAGE

Both the **X server** and the **Font Server** look for "fonts.dir" and "fonts.alias" files in each directory in the font path each time it is set (see xset(1)).

SEE ALSO

X(1), Xserver(1), fs(1), xset(1)

NAME

oclock – display time of day

SYNOPSIS

oclock [-option ...]

DESCRIPTION

Clock simply displays the current time on an analog display

OPTIONS

–fg *foreground color*
: choose a different color for the both hands and the jewel of the clock

–bg *background color*
: choose a different color for the background.

–jewel *jewel color*
: choose a different color for the jewel on the clock.

–minute *minute color*
: choose a different color for the minute hand of the clock.

–hour *hour color*
: choose a different color for the hour hand of the clock.

–backing *{ WhenMapped Always NotUseful }*
: selects an appropriate level of backing store.

–geometry *geometry*
: define the initial window geometry; see *X(1)*.

–display *display*
: specify the display to use; see *X(1)*.

–bd *border color*
: choose a different color for the window border.

–bw *border width*
: choose a different width for the window border. As the Clock widget changes its border around quite a bit, this is most usefully set to zero.

–noshape
: causes the clock to not reshape itself and ancestors to exactly fit the outline of the clock.

–transparent
: causes the clock to consist only of the jewel, the hands, and the border.

COLORS

If you would like your clock to be viewable in color, include the following in the #ifdef COLOR section you read with xrdb:

```
*customization:        -color
```

This will cause oclock to pick up the colors in the app-defaults color customization file: /usr/lib/X11/app-defaults/Clock-color. Below are the default colors:

```
Clock*Background: grey
Clock*BorderColor: light blue
Clock*hour: yellow
Clock*jewel: yellow
Clock*minute: yellow
```

SEE ALSO

X(1), X Toolkit documentation

COPYRIGHT

AUTHOR

Keith Packard, MIT X Consortium

NAME

plbpex – picture level benchmark program

SYNOPSIS

plbpex [-options] verb_file_name(s)

DESCRIPTION

Plbpex is used to view PEX graphics in the GPC, or Graphics Performance Characterization suite. To run it, you must be in the same directory as a verb command, or ∗.vrb file. Some of these are located in the directories demos/gpc/objects, tests, and benchmarks. You must have write permission in that directory because the program will be creating and writing to an error file. Type:

host% ../plbpex [-display geometry bd bg bw hlsr buff] filename

The filenme must be one of the .vrb files located in the objects directory.

OPTIONS

```
-display displaystring      X server to connect to
-geometry geomstring        size and location
-bd color                   border color
-bg color                   background color
-bw pixels                  border width
-hlhsr mode                 one of the following:
        NONE ZBUFF PAINTERS SCANLINE LINE
-buff mode                  one of the following:
                            SINGLE DOUBLE
-h                          this message
```

SEE ALSO

PEX-SI User Guide in doc/extensions/PEX/SI/User_Guide

BUGS

plbpex may crash due to color allocation problems when a window manage is running.

pex only implements hlhsr value none.

COPYRIGHT

NAME

puzzle – 15-puzzle game for X

SYNOPSIS

puzzle [-option ...]

OPTIONS

–display *display*
This option specifies the display to use; see *X(1)*.

–geometry *geometry*
This option specifies the size and position of the puzzle window; see *X(1)*.

–size *WIDTH***x***HEIGHT*
This option specifies the size of the puzzle in squares.

–speed *num*
This option specifies the speed in tiles per second for moving tiles around.

–picture *filename*
This option specifies an image file containing the picture to use on the tiles. Try ‘‘mandrill.cm.’’ This only works on 8-bit pseudo-color screens.

–colormap
This option indicates that the program should create its own colormap for the picture option.

DESCRIPTION

Puzzle with no arguments plays a 4x4 15-puzzle. The control bar has two boxes in it. Clicking in the left box scrambles the puzzle. Clicking in the right box solves the puzzle. Clicking the middle button anywhere else in the control bar causes puzzle to exit. Clicking in the tiled region moves the empty spot to that location if the region you click in is in the same row or column as the empty slot.

SEE ALSO

X(1)

BUGS

The picture option should work on a wider variety of screens.

COPYRIGHT

AUTHOR

Don Bennett, HP Labs

NAME

resize – utility to set TERMCAP and terminal settings to current window size

SYNOPSIS

resize [-u] [–s [row col]]

DESCRIPTION

Resize prints a shell command for setting the TERM and TERMCAP environment variables to indicate the current size of *xterm* window from which the command is run. For this output to take effect, *resize* must either be evaluated as part of the command line (usually done with a shell alias or function) or else redirected to a file which can then be read in. From the C shell (usually known as */bin/csh*), the following alias could be defined in the user's *.cshrc*:

```
% alias rs 'set noglob; `eval resize`'
```

After resizing the window, the user would type:

```
% rs
```

Users of versions of the Bourne shell (usually known as */bin/sh*) that don't have command functions will need to send the output to a temporary file and the read it back in with the ‘‘.’’ command:

```
$ resize >/tmp/out
$ . /tmp/out
```

OPTIONS

The following options may be used with *resize*:

–u This option indicates that Bourne shell commands should be generated even if the user's current shell isn't */bin/sh*.

–c This option indicates that C shell commands should be generated even if the user's current shell isn't */bin/csh*.

–s [*rows columns*]
This option indicates that that Sun console escape sequences will be used instead of the special *xterm* escape code. If *rows* and *columns* are given, *resize* will ask the *xterm* to resize itself. However, the window manager may choose to disallow the change.

FILES

/etc/termcap for the base termcap entry to modify.
˜/.cshrc user's alias for the command.

SEE ALSO

csh(1), tset(1), xterm(1)

AUTHORS

Mark Vandevoorde (MIT-Athena), Edward Moy (Berkeley)
Copyright (c) 1984, 1985 by Massachusetts Institute of Technology.
See *X*(1) for a complete copyright notice.

BUGS

The *-u* or *-c* must appear to the left of *-s* if both are specified.

There should be some global notion of display size; termcap and terminfo need to be rethought in the context of window systems. (Fixed in 4.3BSD, and Ultrix-32 1.2)

NAME

sessreg – manage utmp/wtmp entries for non-init clients

SYNOPSIS

sessreg [-w *wtmp-file*] [-u *utmp-file*] [-l *line-name*] [-h *host-name*] [-s *slot-number*] [-x *Xservers-file*] [-t *ttys-file*] [-a] [-d] *user-name*

DESCRIPTION

Sessreg is a simple program for managing utmp/wtmp entries for xdm sessions.

System V has a better interface to /etc/utmp than BSD; it dynamically allocates entries in the file, instead of writing them at fixed positions indexed by position in /etc/ttys.

To manage BSD-style utmp files, *sessreg* has two strategies. In conjunction with xdm, the -x option counts the number of lines in /etc/ttys and then adds to that the number of the line in the Xservers file which specifies the display. The display name must be specified as the "line-name" using the -l option. This sum is used as the "slot-number" in /etc/utmp that this entry will be written at. In the more general case, the -s option specifies the slot-number directly. If for some strange reason your system uses a file other that /etc/ttys to manage init, the -t option can direct *sessreg* to look elsewhere for a count of terminal sessions.

Conversely, System V managers will not ever need to use these options (-x, -s and -t). To make the program easier to document and explain, *sessreg* accepts the BSD-specific flags in the System V environment and ignores them.

BSD also has a host-name field in the utmp file which doesn't exist in System V. This option is also ignored by the System V version of *sessreg*.

USAGE

In Xstartup, place a call like:

```
sessreg -a -l $DISPLAY -x /usr/lib/X11/xdm/Xservers $USER
```

and in Xreset:

```
sessreg -d -l $DISPLAY -x /usr/lib/X11/xdm/Xservers $USER
```

OPTIONS

-w *wtmp-file*
: This specifies an alternate wtmp file, instead of /usr/adm/wtmp for BSD or /etc/wtmp for sysV. The special name "none" disables writing records to /usr/adm/wtmp.

-u *utmp-file*
: This specifies an alternate utmp file, instead of "/etc/utmp". The special name "none" disables writing records to /etc/utmp.

-l *line-name*
: This describes the "line" name of the entry. For terminal sessions, this is the final pathname segment of the terminal device filename (e.g. ttyd0). For X sessions, it should probably be the local display name given to the users session (e.g. :0). If none is specified, the terminal name will be determined with ttyname(3) and stripped of leading components.

-h *host-name*
: This is set for BSD hosts to indicate that the session was initiated from a remote host. In typical xdm usage, this options is not used.

-s *slot-number*
: Each potential session has a unique slot number in BSD systems, most are identified by the position of the *line-name* in the /etc/ttys file. This option overrides the default position determined with ttyslot(3). This option is inappropriate for use with xdm, the -x option is more useful.

-x *Xservers-file*
As X sessions are one-per-display, and each display is entered in this file, this options sets the *slot-number* to be the number of lines in the *ttys-file* plus the index into this file that the *line-name* is found.

-t *ttys-file*
This specifies an alternate file which the *-x* option will use to count the number of terminal sessions on a host.

-a This session should be added to utmp/wtmp.

-d This session should be deleted from utmp/wtmp. One of -a/-d must be specified.

SEE ALSO

xdm(1)

COPYRIGHT

AUTHOR

Keith Packard, MIT X Consortium

NAME

showfont - font dumper for X font server

SYNOPSIS

showfont [-options ...] [-fn pattern]

DESCRIPTION

Showfont displays data about the font that matches the given *pattern*. The wildcard character "∗" may be used to match any sequence of characters (including none), and "?" to match any single character. If no pattern is given, "∗" is assumed.

The "∗" and "?" characters must be quoted to prevent them from being expanded by the shell.

OPTIONS

–server *host***:***port*
This option specifies the X font server to contact.

–l This option indicates that the bit order of the font is LSBFirst (least significant bit first).

–m This option indicates that the bit order of the font is MSBFirst (most significant bit first).

–L This option indicates that the byte order of the font is LSBFirst (least significant byte first).

–M This option indicates that the byte order of the font is MSBFirst (most significant byte first).

–extents_only
This option indicates that only the font's extents should be displayed.

–start# This option indicates the start of the range of the characters to display (# is a number).

–end# This option indicates the end of the range of the characters to display (# is a number).

–unit# This option specifies the scanline unit of the font (1, 2, 4 or 8). (# is a number).

–pad#

–unit# This option specifies the scanpad unit of the font (1, 2, 4 or 8). (# is a number).

–bitmap_pad#
This option specifies the bitmap padding unit of the font (0, 1, or 2, where 0 is ImageRectMin, 1 is ImageRectMaxWidth and 2 is ImageRectMaxWidth).

SEE ALSO

fs(1), showfont(1), xlsfonts(1)

ENVIRONMENT

FONTSERVER
to get the default host and port to use.

COPYRIGHT

AUTHOR

Dave Lemke, Network Computing Devices, Inc

NAME

showrgb – uncompile an rgb color-name database

SYNOPSIS

showrgb [*database*]

DESCRIPTION

The *showrgb* program reads an rgb color-name database compiled for use with the dbm database routines and converts it back to source form, printing the result to standard output. The default database is the one that X was built with, and may be overridden on the command line. Specify the database name without the *.pag* or *.dir* suffix.

FILES

/usr/lib/X11/rgb default database.

NAME

startx – initialize an X session

SYNOPSIS

startx [[*client*] *options* . . .] [**––** [*server*] *options* . . .]

DESCRIPTION

NOTE: The *startx* script supplied with the X11 distribution is a sample designed more as a base for customization than as a finished product. Site administrators are urged to customize it for their site. And to update this manual page when they do!

The *startx* script is a front end to *xinit* that provides a somewhat nicer user interface for running a single session of the X Window System. It is typically run with no arguments.

To determine the client to run, *startx* first looks for a file called *.xinitrc* in the user's home directory. If that is not found, it uses the file *xinitrc* in the *xinit* library directory. If command line client options are given, they override this behavior. To determine the server to run, *startx* first looks for a file called *.xserverrc* in the user's home directory. If that is not found, it uses the file *xserverrc* in the *xinit* library directory. If command line server options are given, they override this behavior. Users rarely need to provide a *.xserverrc* file. See the *xinit*(1) manual page for more details on the arguments.

The *.xinitrc* is typically a shell script which starts many clients according to the user's preference. When this shell script exits, *startx* kills the server and performs any other session shutdown needed. Most of the clients started by *.xinitrc* should be run in the background. The last client should run in the foreground; when it exits, the session will exit. People often choose a session manager, window manager, or *xterm* as the ''magic'' client.

EXAMPLE

Below is a sample *.xinitrc* that starts several applications and leaves the window manager running as the ''last'' application. Assuming that the window manager has been configured properly, the user then chooses the ''Exit'' menu item to shut down X.

```
xrdb –load $HOME/.Xresources
xsetroot –solid gray &
xbiff –geometry –430+5 &
oclock –geometry 75x75–0–0 &
xload –geometry –80–0 &
xterm –geometry +0+60 –ls &
xterm –geometry +0–100 &
xconsole –geometry –0+0 –fn 5x7 &
exec twm
```

ENVIRONMENT VARIABLES

DISPLAY	This variable gets set to the name of the display to which clients should connect. Note that this gets *set*, not read.

FILES

$(HOME)/.xinitrc	Client to run. Typically a shell script which runs many programs in the background.
$(HOME)/.xserverrc	Server to run. The default is *X*.
/usr/lib/X11/xinit/xinitrc	Client to run if the user has no *.xinitrc* file
/usr/lib/X11/xinit/xserverrc	Client to run if the user has no *.xserverrc* file. This is only needed if the server needs special arguments or is not named *X*.

SEE ALSO

xinit(1)

NAME

twm - Tab Window Manager for the X Window System

SYNTAX

twm [-display *dpy*] [-s] [-f *initfile*] [-v]

DESCRIPTION

Twm is a window manager for the X Window System. It provides titlebars, shaped windows, several forms of icon management, user-defined macro functions, click-to-type and pointer-driven keyboard focus, and user-specified key and pointer button bindings.

This program is usually started by the user's session manager or startup script. When used from *xdm(1)* or *xinit(1)* without a session manager, *twm* is frequently executed in the foreground as the last client. When run this way, exiting *twm* causes the session to be terminated (i.e. logged out).

By default, application windows are surrounded by a ''frame'' with a titlebar at the top and a special border around the window. The titlebar contains the window's name, a rectangle that is lit when the window is receiving keyboard input, and function boxes known as ''titlebuttons'' at the left and right edges of the titlebar.

Pressing pointer Button1 (usually the left-most button unless it has been changed with *xmodmap*) on a titlebutton will invoke the function associated with the button. In the default interface, windows are iconified by clicking (pressing and then immediately releasing) the left titlebutton (which looks like a Dot). Conversely, windows are deiconified by clicking in the associated icon or entry in the icon manager (see description of the variable **ShowIconManager** and of the function **f.showiconmgr**).

Windows are resized by pressing the right titlebutton (which resembles a group of nested squares), dragging the pointer over edge that is to be moved, and releasing the pointer when the outline of the window is the desired size. Similarly, windows are moved by pressing in the title or highlight region, dragging a window outline to the new location, and then releasing when the outline is in the desired position. Just clicking in the title or highlight region raises the window without moving it.

When new windows are created, *twm* will honor any size and location information requested by the user (usually through *-geometry* command line argument or resources for the individual applications). Otherwise, an outline of the window's default size, its titlebar, and lines dividing the window into a 3x3 grid that track the pointer are displayed. Clicking pointer Button1 will position the window at the current position and give it the default size. Pressing pointer Button2 (usually the middle pointer button) and dragging the outline will give the window its current position but allow the sides to be resized as described above. Clicking pointer Button3 (usually the right pointer button) will give the window its current position but attempt to make it long enough to touch the bottom the screen.

OPTIONS

Twm accepts the following command line options:

–display *dpy*
: This option specifies the X server to use.

–s
: This option indicates that only the default screen (as specified by **–display** or by the **DISPLAY** environment variable) should be managed. By default, *twm* will attempt to manage all screens on the display.

–f *filename*
: This option specifies the name of the startup file to use. By default, *twm* will look in the user's home directory for files named *.twmrc.num* (where *num* is a screen number) or *.twmrc*.

–v
: This option indicates that *twm* should print error messages whenever an unexpected X Error event is received. This can be useful when debugging applications but can be distracting in regular use.

CUSTOMIZATION

Much of *twm*'s appearance and behavior can be controlled by providing a startup file in one of the following locations (searched in order for each screen being managed when *twm* begins):

$HOME/.twmrc.*screennumber*
: The *screennumber* is a small positive number (e.g. 0, 1, etc.) representing the screen number (e.g. the last number in the DISPLAY environment variable *host:displaynum.screennum*) that would be used to contact that screen of the display. This is intended for displays with multiple screens of differing visual types.

$HOME/.twmrc
: This is the usual name for an individual user's startup file.

/usr/lib/X11/twm/system.twmrc
: If neither of the preceding files are found, *twm* will look in this file for a default configuration. This is often tailored by the site administrator to provide convenient menus or familiar bindings for novice users.

If no startup files are found, *twm* will use the built-in defaults described above. The only resource used by *twm* is *bitmapFilePath* for a colon-separated list of directories to search when looking for bitmap files (for more information, see the *Athena Widgets* manual and *xrdb(1)*).

Twm startup files are logically broken up into three types of specifications: *Variables*, *Bindings*, *Menus*. The *Variables* section must come first and is used to describe the fonts, colors, cursors, border widths, icon and window placement, highlighting, autoraising, layout of titles, warping, use of the icon manager. The *Bindings* section usually comes second and is used to specify the functions that should be to be invoked when keyboard and pointer buttons are pressed in windows, icons, titles, and frames. The *Menus* section gives any user-defined menus (containing functions to be invoked or commands to be executed).

Variable names and keywords are case-insensitive. Strings must be surrounded by double quote characters (e.g. "blue") and are case-sensitive. A pound sign (#) outside of a string causes the remainder of the line in which the character appears to be treated as a comment.

VARIABLES

Many of the aspects of *twm*'s user interface are controlled by variables that may be set in the user's startup file. Some of the options are enabled or disabled simply by the presence of a particular keyword. Other options require keywords, numbers, strings, or lists of all of these.

Lists are surrounded by braces and are usually separated by whitespace or a newline. For example:

```
AutoRaise { "emacs" "XTerm" "Xmh" }
```

or

```
AutoRaise
{
        "emacs"
        "XTerm"
        "Xmh"
}
```

When a variable containing a list of strings representing windows is searched (e.g. to determine whether or not to enable autoraise as shown above), a string must be an exact, case-sensitive match to the window's name name (given by the WM_NAME window property), resource name or class name (both given by the WM_CLASS window property). The preceding example would enable autoraise on windows named ''emacs'' as well as any *xterm* (since they are of class ''XTerm'') or xmh windows (which are of class ''Xmh'').

String arguments that are interpreted as filenames (see the **Pixmaps**, **Cursors**, and **IconDirectory** below) will prepend the user's directory (specified by the **HOME** environment variable) if the first character is a tilde (~). If, instead, the first character is a colon (:), the name is assumed to refer to one of the internal bitmaps that are used to create the default titlebars symbols: **:xlogo** or **:iconify** (both refer to the X used for the iconify button), **:resize** (the nested squares used by the resize button), and **:question** (the question mark used for non-existent bitmap files).

The following variables may be specified at the top of a *twm* startup file. Lists of Window name prefix strings are indicated by *win-list*. Optional arguments are shown in square brackets:

AutoRaise { *win-list* }
: This variable specifies a list of windows that should automatically be raised whenever the pointer enters the window. This action can be interactively enabled or disabled on individual windows using the function **f.autoraise**.

AutoRelativeResize
: This variable indicates that dragging out a window size (either when initially sizing the window with pointer Button2 or when resizing it) should not wait until the pointer has crossed the window edges. Instead, moving the pointer automatically causes the nearest edge or edges to move by the same amount. This allows the resizing of windows that extend off the edge of the screen. If the pointer is in the center of the window, or if the resize is begun by pressing a titlebutton, *twm* will still wait for the pointer to cross a window edge (to prevent accidents). This option is particularly useful for people who like the press-drag-release method of sweeping out window sizes.

BorderColor *string* [{ *wincolorlist* }]
: This variable specifies the default color of the border to be placed around all non-iconified windows, and may only be given within a **Color** or **Monochrome** list. The optional *wincolorlist* specifies a list of window and color name pairs for specifying particular border colors for different types of windows. For example:

```
BorderColor "gray50"
{
        "XTerm"         "red"
        "xmh"   "green"
}
```

The default is "black".

BorderTileBackground *string* [{ *wincolorlist* }]
: This variable specifies the default background color in the gray pattern used in unhighlighted borders (only if **NoHighlight** hasn't been set), and may only be given within a **Color** or **Monochrome** list. The optional *wincolorlist* allows per-window colors to be specified. The default is "white".

BorderTileForeground *string* [{ *wincolorlist* }]
: This variable specifies the default foreground color in the gray pattern used in unhighlighted borders (only if **NoHighlight** hasn't been set), and may only be given within a **Color** or **Monochrome** list. The optional *wincolorlist* allows per-window colors to be specified. The default is "black".

BorderWidth *pixels*
: This variable specifies the width in pixels of the border surrounding all client window frames if **ClientBorderWidth** has not been specified. This value is also used to set the border size of windows created by *twm* (such as the icon manager). The default is 2.

ButtonIndent *pixels*
: This variable specifies the amount by which titlebuttons should be indented on all sides. Positive values cause the buttons to be smaller than the window text and highlight area so that they stand out. Setting this and the **TitleButtonBorderWidth** variables to 0 makes titlebuttons be as tall and wide as possible. The default is 1.

ClientBorderWidth
: This variable indicates that border width of a window's frame should be set to the initial border width of the window, rather than to the value of **BorderWidth**.

Color { *colors-list* }

This variable specifies a list of color assignments to be made if the default display is capable of displaying more than simple black and white. The *colors-list* is made up of the following color variables and their values: **DefaultBackground**, **DefaultForeground**, **MenuBackground**, **MenuForeground**, **MenuTitleBackground**, **MenuTitleForeground**, and **MenuShadowColor**. The following color variables may also be given a list of window and color name pairs to allow per-window colors to be specified (see **BorderColor** for details): **BorderColor**, **IconManagerHighlight**, **BorderTitleBackground**, **BorderTitleForeground**, **TitleBackground**, **TitleForeground**, **IconBackground**, **IconForeground**, **IconBorderColor**, **IconManagerBackground**, and **IconManagerForeground**. For example:

```
Color
{
        MenuBackground          "gray50"
        MenuForeground      "blue"
        BorderColor             "red" { "XTerm" "yellow" }
        TitleForeground     "yellow"
        TitleBackground     "blue"
}
```

All of these color variables may also be specified for the **Monochrome** variable, allowing the same initialization file to be used on both color and monochrome displays.

ConstrainedMoveTime *milliseconds*

This variable specifies the length of time between button clicks needed to begin a constrained move operation. Double clicking within this amount of time when invoking **f.move** will cause the window only be moved in a horizontal or vertical direction. Setting this value to 0 will disable constrained moves. The default is 400 milliseconds.

Cursors { *cursor-list* }

This variable specifies the glyphs that *twm* should use for various pointer cursors. Each cursor may be defined either from the **cursor** font or from two bitmap files. Shapes from the **cursor** font may be specified directly as:

cursorname "*string*"

where *cursorname* is one of the cursor names listed below, and *string* is the name of a glyph as found in the file /usr/include/X11/cursorfont.h (without the ‘‘XC_’’ prefix). If the cursor is to be defined from bitmap files, the following syntax is used instead:

cursorname "*image*" "*mask*"

The *image* and *mask* strings specify the names of files containing the glyph image and mask in *bitmap(1)* form. The bitmap files are located in the same manner as icon bitmap files. The following example shows the default cursor definitions:

```
Cursors
{
        Frame       "top_left_arrow"
        Title       "top_left_arrow"
        Icon        "top_left_arrow"
        IconMgr"top_left_arrow"
        Move        "fleur"
        Resize      "fleur"
        Menu        "sb_left_arrow"
        Button      "hand2"
        Wait        "watch"
        Select      "dot"
```

```
        Destroy  "pirate"
}
```

DecorateTransients
This variable indicates that transient windows (those containing a WM_TRANSIENT_FOR property) should have titlebars. By default, transients are not reparented.

DefaultBackground *string*
This variable specifies the background color to be used for sizing and information windows. The default is "white".

DefaultForeground *string*
This variable specifies the foreground color to be used for sizing and information windows. The default is "black".

DontIconifyByUnmapping { *win-list* }
This variable specifies a list of windows that should not be iconified by simply unmapping the window (as would be the case if **IconifyByUnmapping** had been set). This is frequently used to force some windows to be treated as icons while other windows are handled by the icon manager.

DontMoveOff
This variable indicates that windows should not be allowed to be moved off the screen. It can be overridden by the **f.forcemove** function.

DontSqueezeTitle [{ *win-list* }]
This variable indicates that titlebars should not be squeezed to their minimum size as described under **SqueezeTitle** below. If the optional window list is supplied, only those windows will be prevented from being squeezed.

ForceIcons
This variable indicates that icon pixmaps specified in the **Icons** variable should override any client-supplied pixmaps.

FramePadding *pixels*
This variable specifies the distance between the titlebar decorations (the button and text) and the window frame. The default is 2 pixels.

IconBackground *string* [{ *win-list* }]
This variable specifies the background color of icons, and may only be specified inside of a **Color** or **Monochrome** list. The optional *win-list* is a list of window names and colors so that per-window colors may be specified. See the **BorderColor** variable for a complete description of the *win-list*. The default is "white".

IconBorderColor *string* [{ *win-list* }]
This variable specifies the color of the border used for icon windows, and may only be specified inside of a **Color** or **Monochrome** list. The optional *win-list* is a list of window names and colors so that per-window colors may be specified. See the **BorderColor** variable for a complete description of the *win-list*. The default is "black".

IconBorderWidth *pixels*
This variable specifies the width in pixels of the border surrounding icon windows. The default is 2.

IconDirectory *string*
This variable specifies the directory that should be searched if if a bitmap file cannot be found in any of the directories in the **bitmapFilePath** resource.

IconFont *string*
This variable specifies the font to be used to display icon names within icons. The default is "variable".

IconForeground *string* [{ *win-list* }]

This variable specifies the foreground color to be used when displaying icons, and may only be specified inside of a **Color** or **Monochrome** list. The optional *win-list* is a list of window names and colors so that per-window colors may be specified. See the **BorderColor** variable for a complete description of the *win-list*. The default is "black".

IconifyByUnmapping [{ *win-list* **}]**

This variable indicates that windows should be iconified by being unmapped without trying to map any icons. This assumes that the user is will remap the window through the icon manager, the **f.warpto** function, or the *TwmWindows* menu. If the optional *win-list* is provided, only those windows will be iconified by simply unmapping. Windows that have both this and the **IconManagerDontShow** options set may not be accessible if no binding to the *TwmWindows* menu is set in the user's startup file.

IconManagerBackground *string* [{ *win-list* }]

This variable specifies the background color to use for icon manager entries, and may only be specified inside of a **Color** or **Monochrome** list. The optional *win-list* is a list of window names and colors so that per-window colors may be specified. See the **BorderColor** variable for a complete description of the *win-list*. The default is "white".

IconManagerDontShow [{ *win-list* }]

This variable indicates that the icon manager should not display any windows. If the optional *win-list* is given, only those windows will not be displayed. This variable is used to prevent windows that are rarely iconified (such as *xclock* or *xload*) from taking up space in the icon manager.

IconManagerFont *string*

This variable specifies the font to be used when displaying icon manager entries. The default is "variable".

IconManagerForeground *string* [{ *win-list* }]

This variable specifies the foreground color to be used when displaying icon manager entries, and may only be specified inside of a **Color** or **Monochrome** list. The optional *win-list* is a list of window names and colors so that per-window colors may be specified. See the **BorderColor** variable for a complete description of the *win-list*. The default is "black".

IconManagerGeometry *string* [*columns*]

This variable specifies the geometry of the icon manager window. The *string* argument is standard geometry specification that indicates the initial full size of the icon manager. The icon manager window is then broken into *columns* pieces and scaled according to the number of entries in the icon manager. Extra entries are wrapped to form additional rows. The default number of columns is 1.

IconManagerHighlight *string* [{ *win-list* }]

This variable specifies the border color to be used when highlighting the icon manager entry that currently has the focus, and can only be specified inside of a **Color** or **Monochrome** list. The optional *win-list* is a list of window names and colors so that per-window colors may be specified. See the **BorderColor** variable for a complete description of the *win-list*. The default is "black".

IconManagers { *iconmgr-list* }

This variable specifies a list of icon managers to create. Each item in the *iconmgr-list* has the following format:

"*winname*" ["*iconname*"] "*geometry*" *columns*

where *winname* is the name of the windows that should be put into this icon manager, *iconname* is the name of that icon manager window's icon, *geometry* is a standard geometry specification, and *columns* is the number of columns in this icon manager as described in **IconManager-**

Geometry. For example:

```
IconManagers
{
        "XTerm"        "=300x5+800+5" 5
        "myhost"       "=400x5+100+5" 2
}
```

Clients whose name or class is ‘‘XTerm’’ will have an entry created in the ‘‘XTerm’’ icon manager. Clients whose name was ‘‘myhost’’ would be put into the ‘‘myhost’’ icon manager.

IconManagerShow { *win-list* }

This variable specifies a list of windows that should appear in the icon manager. When used in conjunction with the **IconManagerDontShow** variable, only the windows in this list will be shown in the icon manager.

IconRegion *geomstring vgrav hgrav gridwidth gridheight*

This variable specifies an area on the root window in which icons are placed if no specific icon location is provided by the client. The *geomstring* is a quoted string containing a standard geometry specification. If more than one **IconRegion** lines are given, icons will be put into the succeeding icon regions when the first is full. The *vgrav* argument should be either **North** or **South** and control and is used to control whether icons are first filled in from the top or bottom of the icon region. Similarly, the *hgrav* argument should be either **East** or **West** and is used to control whether icons should be filled in from left from the right. Icons are laid out within the region in a grid with cells *gridwidth* pixels wide and *gridheight* pixels high.

Icons { *win-list* }

This variable specifies a list of window names and the bitmap filenames that should be used as their icons. For example:

```
Icons
{
        "XTerm"        "xterm.icon"
        "xfd"          "xfd_icon"
}
```

Windows that match ‘‘XTerm’’ and would not be iconified by unmapping, and would try to use the icon bitmap in the file ‘‘xterm.icon’’. If **ForceIcons** is specified, this bitmap will be used even if the client has requested its own icon pixmap.

InterpolateMenuColors

This variable indicates that menu entry colors should be interpolated between entry specified colors. In the example below:

```
Menu "mymenu"
{
        "Title"        ("black":"red")          f.title
        "entry1"                        f.nop
        "entry2"                        f.nop
        "entry3" ("white":"green") f.nop
        "entry4"                        f.nop
        "entry5" ("red":"white")        f.nop
}
```

the foreground colors for ‘‘entry1’’ and ‘‘entry2’’ will be interpolated between black and white, and the background colors between red and green. Similarly, the foreground for ‘‘entry4’’ will be half-way between white and red, and the background will be half-way between green and white.

MakeTitle { *win-list* }
This variable specifies a list of windows on which a titlebar should be placed and is used to request titles on specific windows when **NoTitle** has been set.

MaxWindowSize *string*
This variable specifies a geometry in which the width and height give the maximum size for a given window. This is typically used to restrict windows to the size of the screen. The default is "30000x30000".

MenuBackground *string*
This variable specifies the background color used for menus, and can only be specified inside of a **Color** or **Monochrome** list. The default is "white".

MenuFont *string*
This variable specifies the font to use when displaying menus. The default is "variable".

MenuForeground *string*
This variable specifies the foreground color used for menus, and can only be specified inside of a **Color** or **Monochrome** list. The default is "black".

MenuShadowColor *string*
This variable specifies the color of the shadow behind pull-down menus and can only be specified inside of a **Color** or **Monochrome** list. The default is "black".

MenuTitleBackground *string*
This variable specifies the background color for **f.title** entries in menus, and can only be specified inside of a **Color** or **Monochrome** list. The default is "white".

MenuTitleForeground *string*
This variable specifies the foreground color for **f.title** entries in menus and can only be specified inside of a **Color** or **Monochrome** list. The default is "black".

Monochrome { *colors* }
This variable specifies a list of color assignments that should be made if the screen has a depth of 1. See the description of **Colors**.

MoveDelta *pixels*
This variable specifies the number of pixels the pointer must move before the **f.move** function starts working. Also see the **f.deltastop** function. The default is zero pixels.

NoBackingStore
This variable indicates that *twm*'s menus should not request backing store to minimize repainting of menus. This is typically used with servers that can repaint faster than they can handle backing store.

NoCaseSensitive
This variable indicates that case should be ignored when sorting icon names in an icon manager. This option is typically used with applications that capitalize the first letter of their icon name.

NoDefaults
This variable indicates that *twm* should not supply the default titlebuttons and bindings. This option should only be used if the startup file contains a completely new set of bindings and definitions.

NoGrabServer
This variable indicates that *twm* should not grab the server when popping up menus and moving opaque windows.

NoHighlight [{ *win-list* }]
This variable indicates that borders should not be highlighted to track the location of the pointer. If the optional *win-list* is given, highlighting will only be disabled for those windows. When the border is highlighted, it will be drawn in the current **BorderColor**. When the border is not

highlighted, it will be stippled with an gray pattern using the current **BorderTileForeground** and **BorderTileBackground** colors.

NoIconManagers
This variable indicates that no icon manager should be created.

NoMenuShadows
This variable indicates that menus should not have drop shadows drawn behind them. This is typically used with slower servers since it speeds up menu drawing at the expense of making the menu slightly harder to read.

NoRaiseOnDeiconify
This variable indicates that windows that are deiconified should not be raised.

NoRaiseOnMove
This variable indicates that windows should not be raised when moved. This is typically used to allow windows to slide underneath each other.

NoRaiseOnResize
This variable indicates that windows should not be raised when resized. This is typically used to allow windows to be resized underneath each other.

NoRaiseOnWarp
This variable indicates that windows should not be raised when the pointer is warped into them with the **f.warpto** function. If this option is set, warping to an occluded window may result in the pointer ending up in the occluding window instead the desired window (which causes unexpected behavior with **f.warpring**).

NoSaveUnders
This variable indicates that menus should not request save-unders to minimize window repainting following menu selection. It is typically used with displays that can repaint faster than they can handle save-unders.

NoStackMode [{ *win-list* }]
This variable indicates that client window requests to change stacking order should be ignored. If the optional *win-list* is given, only requests on those windows will be ignored. This is typically used to prevent applications from relentlessly popping themselves to the front of the window stack.

NoTitle [{ *win-list* }]
This variable indicates that windows should not have titlebars. If the optional *win-list* is given, only those windows will not have titlebars. **MakeTitle** may be used with this option to force titlebars to be put on specific windows.

NoTitleFocus
This variable indicates that *twm* should not set keyboard input focus to each window as it is entered. Normally, *twm* sets the focus so that focus and key events from the titlebar and icon managers are delivered to the application. If the pointer is moved quickly and *twm* is slow to respond, input can be directed to the old window instead of the new. This option is typically used to prevent this ‘‘input lag’’ and to work around bugs in older applications that have problems with focus events.

NoTitleHighlight [{ *win-list* }]
This variable indicates that the highlight area of the titlebar, which is used to indicate the window that currently has the input focus, should not be displayed. If the optional *win-list* is given, only those windows will not have highlight areas. This and the **SqueezeTitle** options can be set to substantially reduce the amount of screen space required by titlebars.

OpaqueMove
This variable indicates that the **f.move** function should actually move the window instead of just an outline so that the user can immediately see what the window will look like in the new

position. This option is typically used on fast displays (particularly if **NoGrabServer** is set).

Pixmaps { *pixmaps* }

This variable specifies a list of pixmaps that define the appearance of various images. Each entry is a keyword indicating the pixmap to set, followed by a string giving the name of the bitmap file. The following pixmaps may be specified:

```
Pixmaps
{
        TitleHighlight    "gray1"
}
```

The default for *TitleHighlight* is to use an even stipple pattern.

RandomPlacement

This variable indicates that windows with no specified geometry should should be placed in a pseudo-random location instead of having the user drag out an outline.

ResizeFont *string*

This variable specifies the font to be used for in the dimensions window when resizing windows. The default is "fixed".

RestartPreviousState

This variable indicates that *twm* should attempt to use the WM_STATE property on client windows to tell which windows should be iconified and which should be left visible. This is typically used to try to regenerate the state that the screen was in before the previous window manager was shutdown.

SaveColor { *colors-list* }

This variable indicates a list of color assignments to be stored as pixel values in the root window property _MIT_PRIORITY_COLORS. Clients may elect to preserve these values when installing their own colormap. Note that use of this mechanism is a way an for application to avoid the "technicolor" problem, whereby useful screen objects such as window borders and titlebars disappear when a programs custom colors are installed by the window manager. For example:

```
SaveColor
{
    BorderColor
    TitleBackground
    TitleForeground
    "red"
    "green"
    "blue"
}
```

This would place on the root window 3 pixel values for borders and titlebars, as well as the three color strings, all taken from the default colormap.

ShowIconManager

This variable indicates that the icon manager window should be displayed when *twm* is started. It can always be brought up using the **f.showiconmgr** function.

SortIconManager

This variable indicates that entries in the icon manager should be sorted alphabetically rather than by simply appending new windows to the end.

SqueezeTitle [{ *squeeze-list* }]

This variable indicates that *twm* should attempt to use the SHAPE extension to make titlebars occupy only as much screen space as they need, rather than extending all the way across the top of the window. The optional *squeeze-list* may be used to control the location of the squeezed titlebar along the top of the window. It contains entries of the form:

"*name*" *justification* *num* *denom*

where *name* is a window name, *justification* is either **left**, **center**, or **right**, and *num* and *denom* are numbers specifying a ratio giving the relative position about which the titlebar is justified. The ratio is measured from left to right if the numerator is positive, and right to left if negative. A denominator of 0 indicates that the numerator should be measured in pixels. For convenience, the ratio 0/0 is the same as 1/2 for **center** and -1/1 for **right**. For example:

```
SqueezeTitle
{
    "XTerm"    left      0    0
    "xterm1"   left      1    3
    "xterm2"   left      2    3
    "oclock"   center    0    0
    "emacs"    right     0    0
}
```

The **DontSqueezeTitle** list can be used to turn off squeezing on certain titles.

StartIconified [{ *win-list* }]
This variable indicates that client windows should initially be left as icons until explicitly deiconified by the user. If the optional *win-list* is given, only those windows will be started iconic. This is useful for programs that do not support an *-iconic* command line option or resource.

TitleBackground *string* [{ *win-list* }]
This variable specifies the background color used in titlebars, and may only be specified inside of a **Color** or **Monochrome** list. The optional *win-list* is a list of window names and colors so that per-window colors may be specified. The default is "white".

TitleButtonBorderWidth *pixels*
This variable specifies the width in pixels of the border surrounding titlebuttons. This is typically set to 0 to allow titlebuttons to take up as much space as possible and to not have a border. The default is 1.

TitleFont *string*
This variable specifies the font to used for displaying window names in titlebars. The default is "variable".

TitleForeground *string* [{ *win-list* }]
This variable specifies the foreground color used in titlebars, and may only be specified inside of a **Color** or **Monochrome** list. The optional *win-list* is a list of window names and colors so that per-window colors may be specified. The default is "black".

TitlePadding *pixels*
This variable specifies the distance between the various buttons, text, and highlight areas in the titlebar. The default is 8 pixels.

UnknownIcon *string*
This variable specifies the filename of a bitmap file to be used as the default icon. This bitmap will be used as the icon of all clients which do not provide an icon bitmap and are not listed in the **Icons** list.

UsePPosition *string*
This variable specifies whether or not *twm* should honor program-requested locations (given by the **PPosition** flag in the WM_NORMAL_HINTS property) in the absence of a user-specified position. The argument *string* may have one of three values: **"off"** (the default) indicating that *twm* should ignore the program-supplied position, **"on"** indicating that the position should be

used, and **"non-zero"** indicating that the position should used if it is other than (0,0). The latter option is for working around a bug in older toolkits.

WarpCursor [{ *win-list* }]
This variable indicates that the pointer should be warped into windows when they are deiconified. If the optional *win-list* is given, the pointer will only be warped when those windows are deiconified.

WindowRing { *win-list* }
This variable specifies a list of windows along which the **f.warpring** function cycles.

WarpUnmapped
This variable indicates that that the **f.warpto** function should deiconify any iconified windows it encounters. This is typically used to make a key binding that will pop a particular window (such as *xmh*), no matter where it is. The default is for **f.warpto** to ignore iconified windows.

XorValue *number*
This variable specifies the value to use when drawing window outlines for moving and resizing. This should be set to a value that will result in a variety of of distinguishable colors when exclusive-or'ed with the contents of the user's typical screen. Setting this variable to 1 often gives nice results if adjacent colors in the default colormap are distinct. By default, *twm* will attempt to cause temporary lines to appear at the opposite end of the colormap from the graphics.

Zoom [*count*]
This variable indicates that outlines suggesting movement of a window to and from its iconified state should be displayed whenever a window is iconified or deiconified. The optional *count* argument specifies the number of outlines to be drawn. The default count is 8.

The following variables must be set after the fonts have been assigned, so it is usually best to put them at the end of the variables or beginning of the bindings sections:

DefaultFunction *function*
This variable specifies the function to be executed when a key or button event is received for which no binding is provided. This is typically bound to **f.nop**, **f.beep**, or a menu containing window operations.

WindowFunction *function*
This variable specifies the function to execute when a window is selected from the **TwmWindows** menu. If this variable is not set, the window will be deiconified and raised.

BINDINGS

After the desired variables have been set, functions may be attached titlebuttons and key and pointer buttons. Titlebuttons may be added from the left or right side and appear in the titlebar from left-to-right according to the order in which they are specified. Key and pointer button bindings may be given in any order.

Titlebuttons specifications must include the name of the pixmap to use in the button box and the function to be invoked when a pointer button is pressed within them:

LeftTitleButton "*bitmapname*" = *function*

or

RightTitleButton "*bitmapname*" = *function*

The *bitmapname* may refer to one of the built-in bitmaps (which are scaled to match **TitleFont**) by using the appropriate colon-prefixed name described above.

Key and pointer button specifications must give the modifiers that must be pressed, over which parts of the screen the pointer must be, and what function is to be invoked. Keys are given as strings containing the appropriate keysym name; buttons are given as the keywords **Button1**-**Button5**:

"FP1" = *modlist* : *context* : *function*
Button1 = *modlist* : *context* : *function*

The *modlist* is any combination of the modifier names **shift**, **control**, **lock**, **meta**, **mod1**, **mod2**, **mod3**, **mod4**, or **mod5** (which may be abbreviated as **s**, **c**, **l**, **m**, **m1**, **m2**, **m3**, **m4**, **m5**, respectively) separated by a vertical bar (|). Similarly, the *context* is any combination of **window**, **title**, **icon**, **root**, **frame**, **iconmgr**, their first letters (**iconmgr** abbreviation is **m**), or **all**, separated by a vertical bar. The *function* is any of the **f.** keywords described below. For example, the default startup file contains the following bindings:

```
Button1 =      : root           : f.menu "TwmWindows"
Button1 = m    : window | icon  : f.function "move-or-lower"
Button2 = m    : window | icon  : f.iconify
Button3 = m    : window | icon  : f.function "move-or-raise"
Button1 =      : title          : f.function "move-or-raise"
Button2 =      : title          : f.raiselower
Button1 =      : icon           : f.function "move-or-iconify"
Button2 =      : icon           : f.iconify
Button1 =      : iconmgr        : f.iconify
Button2 =      : iconmgr        : f.iconify
```

A user who wanted to be able to manipulate windows from the keyboard could use the following bindings:

```
"F1"       =        : all      : f.iconify
"F2"       =        : all      : f.raiselower
"F3"       =        : all      : f.warpring "next"
"F4"       =        : all      : f.warpto "xmh"
"F5"       =        : all      : f.warpto "emacs"
"F6"       =        : all      : f.colormap "next"
"F7"       =        : all      : f.colormap "default"
"F20"      =        : all      : f.warptoscreen "next"
"Left"     = m      : all      : f.backiconmgr
"Right"    = m | s  : all      : f.forwiconmgr
"Up"       = m      : all      : f.upiconmgr
"Down"     = m | s  : all      : f.downiconmgr
```

Twm provides many more window manipulation primitives than can be conveniently stored in a titlebar, menu, or set of key bindings. Although a small set of defaults are supplied (unless the **NoDefaults** is specified), most users will want to have their most common operations bound to key and button strokes. To do this, *twm* associates names with each of the primitives and provides *user-defined functions* for building higher level primitives and *menus* for interactively selecting among groups of functions.

User-defined functions contain the name by which they are referenced in calls to **f.function** and a list of other functions to execute. For example:

```
Function "move-or-lower" { f.move f.deltastop f.lower }
Function "move-or-raise" { f.move f.deltastop f.raise }
Function "move-or-iconify" { f.move f.deltastop f.iconify }
Function "restore-colormap" { f.colormap "default" f.lower }
```

The function name must be used in **f.function** exactly as it appears in the function specification.

In the descriptions below, if the function is said to operate on the selected window, but is invoked from a root menu, the cursor will be changed to the **Select** cursor and the next window to receive a button press will be chosen:

! *string* This is an abbreviation for **f.exec** *string*.

f.autoraise

This function toggles whether or not the selected window is raised whenever entered by the pointer. See the description of the variable **AutoRaise**.

f.backiconmgr
This function warps the pointer to the previous column in the current icon manager, wrapping back to the previous row if necessary.

f.beep This function sounds the keyboard bell.

f.bottomzoom
This function is similar to the **f.fullzoom** function, but resizes the window to fill only the bottom half of the screen.

f.circledown
This function lowers the top-most window that occludes another window.

f.circleup
This function raises the bottom-most window that is occluded by another window.

f.colormap *string*
This function rotates the colormaps (obtained from the WM_COLORMAP_WINDOWS property on the window) that *twm* will display when the pointer is in this window. The argument *string* may have one of the following values: **"next"**, **"prev"**, and **"default"**. It should be noted here that in general, the installed colormap is determined by keyboard focus. A pointer driven keyboard focus will install a private colormap upon entry of the window owning the colormap. Using the click to type model, private colormaps will not be installed until the user presses a mouse button on the target window.

f.deiconify
This function deiconifies the selected window. If the window is not an icon, this function does nothing.

f.delete This function sends the WM_DELETE_WINDOW message to the selected window if the client application has requested it through the WM_PROTOCOLS window property. The application is supposed to respond to the message by removing the indicated window. If the window has not requested WM_DELETE_WINDOW messages, the keyboard bell will be rung indicating that the user should choose an alternative method. Note this is very different from f.destroy. The intent here is to delete a single window, not necessarily the entire application.

f.deltastop
This function allows a user-defined function to be aborted if the pointer has been moved more than *MoveDelta* pixels. See the example definition given for **Function "move-or-raise"** at the beginning of the section.

f.destroy
This function instructs the X server to close the display connection of the client that created the selected window. This should only be used as a last resort for shutting down runaway clients. See also f.delete.

f.downiconmgr
This function warps the pointer to the next row in the current icon manger, wrapping to the beginning of the next column if necessary.

f.exec *string*
This function passes the argument *string* to /bin/sh for execution. In multiscreen mode, if *string* starts a new X client without giving a display argument, the client will appear on the screen from which this function was invoked.

f.focus This function toggles the keyboard focus of the server to the selected window, changing the focus rule from pointer-driven if necessary. If the selected window already was focused, this function executes an **f.unfocus**.

f.forcemove
This function is like **f.move** except that it ignores the **DontMoveOff** variable.

f.forwiconmgr
This function warps the pointer to the next column in the current icon manager, wrapping to the beginning of the next row if necessary.

f.fullzoom
This function resizes the selected window to the full size of the display or else restores the original size if the window was already zoomed.

f.function *string*
This function executes the user-defined function whose name is specified by the argument *string*.

f.hbzoom
This function is a synonym for **f.bottomzoom**.

f.hideiconmgr
This function unmaps the current icon manager.

f.horizoom
This variable is similar to the **f.zoom** function except that the selected window is resized to the full width of the display.

f.htzoom
This function is a synonym for **f.topzoom**.

f.hzoom This function is a synonym for **f.horizoom**.

f.iconify This function iconifies or deiconifies the selected window or icon, respectively.

f.identify
This function displays a summary of the name and geometry of the selected window. Clicking the pointer or pressing a key in the window will dismiss it.

f.lefticonmgr
This function similar to **f.backiconmgr** except that wrapping does not change rows.

f.leftzoom
This variable is similar to the **f.bottomzoom** function but causes the selected window is only resized to the left half of the display.

f.lower This function lowers the selected window.

f.menu *string*
This function invokes the menu specified by the argument *string*. Cascaded menus may be built by nesting calls to **f.menu**.

f.move This function drags an outline of the selected window (or the window itself if the **OpaqueMove** variable is set) until the invoking pointer button is released. Double clicking within the number of milliseconds given by **ConstrainedMoveTime** warps the pointer to the center of the window and constrains the move to be either horizontal or vertical depending on which grid line is crossed. To abort a move, press another button before releasing the first button.

f.nexticonmgr
This function warps the pointer to the next icon manager containing any windows on the current or any succeeding screen.

f.nop This function does nothing and is typically used with the **DefaultFunction** or **WindowFunction** variables or to introduce blank lines in menus.

f.previconmgr
This function warps the pointer to the previous icon manager containing any windows on the current or preceding screens.

f.quit This function causes *twm* to restore the window's borders and exit. If *twm* is the first client invoked from *xdm*, this will result in a server reset.

f.raise This function raises the selected window.

f.raiselower
This function raises the selected window to the top of the stacking order if it is occluded by any windows, otherwise the window will be lowered.

f.refresh This function causes all windows to be refreshed.

f.resize This function displays an outline of the selected window. Crossing a border (or setting **AutoRelativeResize**) will cause the outline to begin to rubber band until the invoking button is released. To abort a resize, press another button before releasing the first button.

f.restart This function kills and restarts *twm*.

f.righticonmgr
This function is similar to **f.nexticonmgr** except that wrapping does not change rows.

f.rightzoom
This variable is similar to the **f.bottomzoom** function except that the selected window is only resized to the right half of the display.

f.saveyourself
This function sends a WM_SAVEYOURSELF message to the selected window if it has requested the message in its WM_PROTOCOLS window property. Clients that accept this message are supposed to checkpoint all state associated with the window and update the WM_COMMAND property as specified in the ICCCM. If the selected window has not selected for this message, the keyboard bell will be rung.

f.showiconmgr
This function maps the current icon manager.

f.sorticonmgr
This function sorts the entries in the current icon manager alphabetically. See the variable **SortIconManager**.

f.title This function provides a centered, unselectable item in a menu definition. It should not be used in any other context.

f.topzoom
This variable is similar to the **f.bottomzoom** function except that the selected window is only resized to the top half of the display.

f.unfocus
This function resets the focus back to pointer-driven. This should be used when a focused window is no longer desired.

f.upiconmgr
This function warps the pointer to the previous row in the current icon manager, wrapping to the last row in the same column if necessary.

f.vlzoom This function is a synonym for **f.leftzoom**.

f.vrzoom
This function is a synonym for **f.rightzoom**.

f.warpring *string*
This function warps the pointer to the next or previous window (as indicated by the argument *string*, which may be **"next"** or **"prev"**) specified in the **WindowRing** variable.

f.warpto *string*
This function warps the pointer to the window which has a name or class that matches *string*. If

the window is iconified, it will be deiconified if the variable **WarpUnmapped** is set or else ignored.

f.warptoiconmgr *string*
This function warps the pointer to the icon manager entry associated with the window containing the pointer in the icon manager specified by the argument *string*. If *string* is empty (i.e. ""), the current icon manager is chosen.

f.warptoscreen *string*
This function warps the pointer to the screen specified by the argument *string*. *String* may be a number (e.g. **"0"** or **"1"**), the word **"next"** (indicating the current screen plus 1, skipping over any unmanaged screens), the word **"back"** (indicating the current screen minus 1, skipping over any unmanaged screens), or the word **"prev"** (indicating the last screen visited.

f.winrefresh
This function is similar to the **f.refresh** function except that only the selected window is refreshed.

f.zoom This function is similar to the **f.fullzoom** function, except that the only the height of the selected window is changed.

MENUS

Functions may be grouped and interactively selected using pop-up (when bound to a pointer button) or pull-down (when associated with a titlebutton) menus. Each menu specification contains the name of the menu as it will be referred to by **f.menu**, optional default foreground and background colors, the list of item names and the functions they should invoke, and optional foreground and background colors for individual items:

```
Menu "menuname" [ ("deffore":"defback") ]
{
        string1   [ ("fore1":"backn")]         function1
        string2   [ ("fore2":"backn")]         function2
                  .
                  .
                  .
        stringN   [ ("foreN":"backN")]         functionN
}
```

The *menuname* is case-sensitive. The optional *deffore* and *defback* arguments specify the foreground and background colors used on a color display to highlight menu entries. The *string* portion of each menu entry will be the text which will appear in the menu. The optional *fore* and *back* arguments specify the foreground and background colors of the menu entry when the pointer is not in the entry. These colors will only be used on a color display. The default is to use the colors specified by the **MenuForeground** and **MenuBackground** variables. The *function* portion of the menu entry is one of the functions, including any user-defined functions, or additional menus.

There is a special menu named **TwmWindows** which contains the names of all of the client and *twm*-supplied windows. Selecting an entry will cause the **WindowFunction** to be executed on that window. If **WindowFunction** hasn't been set, the window will be deiconified and raised.

ICONS

Twm supports several different ways of manipulating iconified windows. The common pixmap-and-text style may be laid out by hand or automatically arranged as described by the **IconRegion** variable. In addition, a terse grid of icon names, called an icon manager, provides a more efficient use of screen space as well as the ability to navigate among windows from the keyboard.

An icon manager is a window that contains names of selected or all windows currently on the display. In addition to the window name, a small button using the default iconify symbol will be displayed to the left of the name when the window is iconified. By default, clicking on an entry in the icon manager performs

f.iconify. To change the actions taken in the icon manager, use the the **iconmgr** context when specifying button and keyboard bindings.

Moving the pointer into the icon manager also directs keyboard focus to the indicated window (setting the focus explicitly or else sending synthetic events **NoTitleFocus** is set). Using the **f.upiconmgr**, **f.downiconmgr f.lefticonmgr**, and **f.righticonmgr** functions, the input focus can be changed between windows directly from the keyboard.

BUGS

The resource manager should have been used instead of all of the window lists.

The **IconRegion** variable should take a list.

Double clicking very fast to get the constrained move function will sometimes cause the window to move, even though the pointer is not moved.

If **IconifyByUnmapping** is on and windows are listed in **IconManagerDontShow** but not in **DontIconifyByUnmapping**, they may be lost if they are iconified and no bindings to **f.menu "TwmWindows"** or **f.warpto** are setup.

FILES

$HOME/.twmrc.<screen number>
$HOME/.twmrc
/usr/lib/X11/twm/system.twmrc

ENVIRONMENT VARIABLES

DISPLAY
: This variable is used to determine which X server to use. It is also set during **f.exec** so that programs come up on the proper screen.

HOME
: This variable is used as the prefix for files that begin with a tilde and for locating the *twm* startup file.

SEE ALSO

X(1), Xserver(1), xdm(1), xrdb(1)

COPYRIGHT

AUTHORS

Tom LaStrange, Solbourne Computer; Jim Fulton, MIT X Consortium; Steve Pitschke, Stardent Computer; Keith Packard, MIT X Consortium; Dave Sternlicht, MIT X Consortium; Dave Payne, Apple Computer.

NAME

viewres - graphical class browser for Xt

SYNOPSIS

viewres [-option ...]

DESCRIPTION

The *viewres* program displays a tree showing the widget class hierarchy of the Athena Widget Set. Each node in the tree can be expanded to show the resources that the corresponding class adds (i.e. does not inherit from its parent) when a widget is created. This application allows the user to visually examine the structure and inherited resources for the Athena Widget Set.

OPTIONS

Viewres accepts all of the standard toolkit command line options as well as the following:

–top *name*
: This option specifies the name of the highest widget in the hierarchy to display. This is typically used to limit the display to a subset of the tree. The default is *Object*.

–variable
: This option indicates that the widget variable names (as declared in header files) should be displayed in the nodes rather than the widget class name. This is sometimes useful to distinguish widget classes that share the same name (such as *Text*).

–vertical
: This option indicates that the tree should be displayed top to bottom rather left to right.

VIEW MENU

The way in which the tree is displayed may be changed through the entries in the **View** menu:

Show Variable Names
: This entry causes the node labels to be set to the variable names used to declare the corresponding widget class. This operation may also be performed with the **SetLabelType(variable)** translation.

Show Class Names
: This entry causes the node labels to be set to the class names used when specifying resources. This operation may also be performed with the **SetLabelType(class)** translation.

Layout Horizontal
: This entry causes the tree to be laid out from left to right. This operation may also be performed with the *SetOrientation(West)* translation.

Layout Vertical
: This entry causes the tree to be laid out from top to bottom. This operation may also be performed with the *SetOrientation(North)* translation.

Show Resource Boxes
: This entry expands the selected nodes (see next section) to show the new widget and constraint resources. This operation may also be performed with the *Resources(on)* translation.

Hide Resource Boxes
: This entry removes the resource displays from the selected nodes (usually to conserve space). This operation may also be performed with the *Resources(off)* translation.

SELECT MENU

Resources for a single widget class can be displayed by clicking **Button2** on the corresponding node, or by adding the node to the selection list with **Button1** and using the **Show Resource Boxes** entry in the **View** menu. Since **Button1** actually toggles the selection state of a node, clicking on a selected node will cause it to be removed from the selected list.

Collections of nodes may also be selected through the various entries in the **Select** menu:

Unselect All
This entry removes all nodes from the selection list. This operation may also be performed with the *Select(nothing)* translation.

Select All
This entry adds all nodes to the selection list. This operation may also be performed with the *Select(all)* translation.

Invert All
This entry adds unselected nodes to, and removes selected nodes from, the selection list. This operation may also be performed with the *Select(invert)* translation.

Select Parent
This entry selects the immediate parents of all selected nodes. This operation may also be performed with the *Select(parent)* translation.

Select Ancestors
This entry recursively selects all parents of all selected nodes. This operation may also be performed with the *Select(ancestors)* translation.

Select Children
This entry selects the immediate children of all selected nodes. This operation may also be performed with the *Select(children)* translation.

Select Descendants
This entry recursively selects all children of all selected nodes. This operation may also be performed with the *Select(descendants)* translation.

Select Has Resources
This entry selects all nodes that add new resources (regular or constraint) to their corresponding widget classes. This operation may also be performed with the *Select(resources)* translation.

Select Shown Resource Boxes
This entry selects all nodes whose resource boxes are currently expanded (usually so that they can be closed with **Hide Resource Boxes**). This operation may also be performed with the *Select(shown)* translation.

ACTIONS

The following application actions are provided:

Quit()
This action causes *viewres* to exit.

SetLabelType(*type***)**
This action sets the node labels to display the widget *variable* or *class* names, according to the argument *type*.

SetOrientation(*direction***)**
This action sets the root of the tree to be one of the following areas of the window: *West*, *North*, *East*, or *South*.

Select(*what***)**
This action selects the indicated nodes, as described in the **VIEW MENU** section: *nothing* (unselects all nodes), *invert*, *parent*, *ancestors*, *children*, *descendants*, *resources*, *shown*.

Resources(*op***)**
This action turns *on*, *off*, or *toggles* the resource boxes for the selected nodes. If invoked from within one of the nodes (through the keyboard or pointer), only that node is used.

WIDGET HIERARCHY

Resources may be specified for the following widgets:

```
Viewres viewres
    Paned pane
        Box buttonbox
            Command quit
            MenuButton view
                SimpleMenu viewMenu
                    SmeBSB layoutHorizontal
                    SmeBSB layoutVertical
                    SmeLine line1
                    SmeBSB namesVariable
                    SmeBSB namesClass
                    SmeLine line2
                    SmeBSB viewResources
                    SmeBSB viewNoResources
            MenuButton select
                SimpleMenu selectMenu
                    SmeBSB unselect
                    SmeBSB selectAll
                    SmeBSB selectInvert
                    SmeLine line1
                    SmeBSB selectParent
                    SmeBSB selectAncestors
                    SmeBSB selectChildren
                    SmeBSB selectDescendants
                    SmeLine line2
                    SmeBSB selectHasResources
                    SmeBSB selectShownResources
        Form treeform
            Porthole porthole
                Tree tree
                    Box variable-name
                        Toggle variable-name
                        List variable-name
            Panner panner
```

where *variable-name* is the widget variable name of each node.

SEE ALSO

X(1), xrdb(1), listres(1), editres(1), appres(1), appropriate widget documents

COPYRIGHT

AUTHOR

Jim Fulton, MIT X Consortium

NAME

x11perf – X11 server performance test program

SYNTAX

x11perf [–option ...]

DESCRIPTION

The *x11perf* program runs one or more performance tests and reports how fast an X server can execute the tests.

Many graphics benchmarks assume that the graphics device is used to display the output of a single fancy graphics application, and that the user gets his work done on some other device, like a terminal. Such benchmarks usually measure drawing speed for lines, polygons, text, etc.

Since workstations are not used as standalone graphics engines, but as super-terminals, *x11perf* measures window management performance as well as traditional graphics performace. *x11perf* includes benchmarks for the time it takes to create and map windows (as when you start up an application); to map a pre-existing set of windows onto the screen (as when you deiconify an application or pop up a menu); and to rearrange windows (as when you slosh windows to and fro trying to find the one you want).

x11perf also measures graphics performance for operations not normally used in standalone graphics displays, but are nonetheless used frequently by X applications. Such operations include CopyPlane (used to map bitmaps into pixels), scrolling (used in text windows), and various stipples and tiles (used for CAD and color half-toning, respectively).

x11perf should be used to analyze particular strengths and weaknesses of servers, and is most useful to a server writer who wants to analyze and improve a server. *x11perf* is meant to comprehensively exercise just about every X11 operation you can perform; it does not purport to be a representative sample of the operations that X11 applications actually use. While it can be used as a benchmark, it was written and is intended as a performance testing tool.

As such, *x11perf* DOES NOT whittle down measurements to a single ‘‘HeXStones’’ or ‘‘MeXops’’ number. We consider such numbers to be uninformative at best and misleading at worst. Some servers which are very fast for certain applications can be very slow for others. No single number or small set of numbers are sufficient to characterize how an X implementation will perform on all applications. However, by knowledge of your favorite application, you may be able to use the numbers *x11perf* reports to predict its performance on a given X implementation.

That said, you might also want to look at *x11perfcomp(1),* a program to compare the outputs of different *x11perf* runs. You provide a list of files containing results from *x11perf,* and it lays them out in a nice tabular format.

For repeatable results, *x11perf* should be run using a local connection on a freshly-started server. The default configuration runs each test 5 times, in order to see if each trial takes approximately the same amount of time. Strange glitches should be examined; if non-repeatable one might chalk them up to daemons and network traffic. Each trial is run for 5 seconds, in order to reduce random time differences. The number of objects processed per second is displayed to 3 significant digits, but you’ll be lucky on most UNIX system if the numbers are actually consistent to 2 digits. *x11perf* moves the cursor out of the test window; you should be careful not to bump the mouse and move it back into the window. (A prize to people who correctly explain why!!).

Before running a test, *x11perf* determines what the round trip time to the server is, and factors this out of the final timing reported. It ensures that the server has actually performed the work requested by fetching a pixel back from the test window, which means that servers talking to graphics accelerators can’t claim that they are done, while in the meantime the accelerator is painting madly.

By default *x11perf* automatically calibrates the number of repetitions of each test, so that each should take approximately the same length of time to run across servers of widely differing speeds. However, since each test must be run to completion at least once, some slow servers may take a very long time, particularly on the window moving and resizing tests, and on the arc drawing tests.

All timing reports are for the smallest object involved. For example, the line tests use a PolyLine request to paint several lines at once, but report how many lines per second the server can paint, not how many PolyLine requests per second. Text tests paint a line of characters, but report on the number of characters per second. Some window tests map, unmap, or move a single parent window, but report on how many children windows per second the server can map, unmap, or move.

The current program is mostly the responsibility of Joel McCormack. It is based upon the x11perf developed by Phil Karlton, Susan Angebranndt, and Chris Kent, who wanted to assess performance differences between various servers. Mary Walker, Todd Newman, and several tests were added in order to write and tune the PMAX (DECStation 3100) servers. For a general release to the world, *x11perf* was rewritten to ease making comparisons between widely varying machines, to cover most important (and unimportant) X functionality, and to exercise graphics operations in as many different orientations and alignments as possible.

OPTIONS

x11perf is solely Xlib based, and accepts the options listed below:

–display host:dpy
Specifies which display to use.

–sync Runs the tests in synchronous mode. Normally only useful for debugging *x11perf* .

–pack Runs rectangle tests so that they pack rectangles right next to each other. This makes it easy to debug server code for stipples and tiles...if the pattern looks ugly, you’ve got alignment problems.

–repeat <*n*> Repeats each test *n* times (by default each test is run 5 times).

–time <*s*> Specifies how long in seconds each test should be run (default 5 seconds).

–all Runs all tests. This may take a while.

–range <*test1*>**[,**<*test2*>**]**
Runs all the tests starting from the specified name *test1* until the name *test2*, including both the specified tests. The testnames should be one of the options starting from -dot. (eg) -range line100 will peform the tests from the 100 pixel line test, and go on till the last test, -range line100,dline10 will do the tests from line100 to dline10.

–labels Generates just the descriptive labels for each test specified. See *x11perfcomp* for more details.

–fg *color* Specifies the foreground color to use.

–bg *color* Specifies the background color to use.

–rop <rop0 rop1 ...>
Use specified raster ops (default is GXcopy). This option only affects graphics benchmarks in which the graphics function is actually used.

–pm <pm0 pm1 ...>
Use specified planemasks (default is ˜0). This option only affects graphics benchmarks in which the planemask is actually used.

–depth <depth>
Use a visual with <depth> planes per pixel (default is the default visual).

–dot Dot.

–rect1 1x1 solid-filled rectangle.

–rect10 10x10 solid-filled rectangle.

–rect100 100x100 solid-filled rectangle.

–rect500 500x500 solid-filled rectangle.

–srect1 1x1 transparent stippled rectangle, 8x8 stipple pattern.

–srect10 10x10 transparent stippled rectangle, 8x8 stipple pattern.

–srect100 100x100 transparent stippled rectangle, 8x8 stipple pattern.

–srect500 500x500 transparent stippled rectangle, 8x8 stipple pattern.

–osrect1 1x1 opaque stippled rectangle, 8x8 stipple pattern.

–osrect10 10x10 opaque stippled rectangle, 8x8 stipple pattern.

–osrect100 100x100 opaque stippled rectangle, 8x8 stipple pattern.

–osrect500 500x500 opaque stippled rectangle, 8x8 stipple pattern.

–tilerect1 1x1 tiled rectangle, 4x4 tile pattern.

–tilerect10 10x10 tiled rectangle, 4x4 tile pattern.

–tilerect100 100x100 tiled rectangle, 4x4 tile pattern.

–tilerect500 500x500 tiled rectangle, 4x4 tile pattern.

–bigsrect1 1x1 stippled rectangle, 161x145 stipple pattern.

–bigsrect10 10x10 stippled rectangle, 161x145 stipple pattern.

–bigsrect100 100x100 stippled rectangle, 161x145 stipple pattern.

–bigsrect500 500x500 stippled rectangle, 161x145 stipple pattern.

–bigosrect1 1x1 opaque stippled rectangle, 161x145 stipple pattern.

–bigosrect10 10x10 opaque stippled rectangle, 161x145 stipple pattern.

–bigosrect100 100x100 opaque stippled rectangle, 161x145 stipple pattern.

–bigosrect500 500x500 opaque stippled rectangle, 161x145 stipple pattern.

–bigtilerect1 1x1 tiled rectangle, 161x145 tile pattern.

–bigtilerect10 10x10 tiled rectangle, 161x145 tile pattern.

–bigtilerect100 100x100 tiled rectangle, 161x145 tile pattern.

–bigtilerect500 500x500 tiled rectangle, 161x145 tile pattern.

–eschertilerect1 1x1 tiled rectangle, 215x208 tile pattern.

–eschertilerect10
10x10 tiled rectangle, 215x208 tile pattern.

–eschertilerect100
100x100 tiled rectangle, 215x208 tile pattern.

–eschertilerect500
500x500 tiled rectangle, 215x208 tile pattern.

–seg1 1-pixel thin line segment.

–seg10 10-pixel thin line segment.

–seg100 100-pixel thin line segment.

–seg500 500-pixel thin line segment.

–seg100c1 100-pixel thin line segment (1 obscuring rectangle).

–seg100c2 100-pixel thin line segment (2 obscuring rectangles).

–seg100c3 100-pixel thin line segment (3 obscuring rectangles).

–dseg10 10-pixel thin dashed segment (3 on, 2 off).

–dseg100 100-pixel thin dashed segment (3 on, 2 off).

–ddseg100 100-pixel thin double-dashed segment (3 fg, 2 bg).

–hseg10 10-pixel thin horizontal line segment.

–hseg100 100-pixel thin horizontal line segment.

–hseg500 500-pixel thin horizontal line segment.

–vseg10 10-pixel thin vertical line segment.

–vseg100 100-pixel thin vertical line segment.

–vseg500 500-pixel thin vertical line segment.

–whseg10 10-pixel wide horizontal line segment.

–whseg100 100-pixel wide horizontal line segment.

–whseg500 500-pixel wide horizontal line segment.

–wvseg10 10-pixel wide vertical line segment.

–wvseg100 100-pixel wide vertical line segment.

–wvseg500 500-pixel wide vertical line segment.

–line1 1-pixel thin (width 0) line.

–line10 10-pixel thin line.

–line100 100-pixel thin line.

–line500 500-pixel thin line.

–dline10 10-pixel thin dashed line (3 on, 2 off).

–dline100 100-pixel thin dashed line (3 on, 2 off).

–ddline100 100-pixel thin double-dashed line (3 fg, 2 bg).

–wline10 10-pixel line, line width 1.

–wline100 100-pixel line, line width 10.

–wline500 500-pixel line, line width 50.

–wdline100 100-pixel dashed line, line width 10 (30 on, 20 off).

–wddline100 100-pixel double-dashed line, line width 10 (30 fg, 20 bg).

–orect10 10x10 thin rectangle outline.

–orect100 100-pixel thin vertical line segment.

–orect500 500-pixel thin vertical line segment.

–worect10 10x10 wide rectangle outline.

–worect100 100-pixel wide vertical line segment.

–worect500 500-pixel wide vertical line segment.

–circle1 1-pixel diameter thin (line width 0) circle.

–circle10 10-pixel diameter thin circle.

–circle100 100-pixel diameter thin circle.

–circle500 500-pixel diameter thin circle.

–dcircle100 100-pixel diameter thin dashed circle (3 on, 2 off).

–ddcircle100 100-pixel diameter thin double-dashed circle (3 fg, 2 bg).

–wcircle10 10-pixel diameter circle, line width 1.

–wcircle100 100-pixel diameter circle, line width 10.

–wcircle500 500-pixel diameter circle, line width 50.

–wdcircle100 100-pixel diameter dashed circle, line width 10 (30 on, 20 off).

–wddcircle100 100-pixel diameter double-dashed circle, line width 10 (30 fg, 20 bg).

–pcircle10 10-pixel diameter thin partial circle, orientation and arc angle evenly distributed.

–pcircle100 100-pixel diameter thin partial circle.

–wpcircle10 10-pixel diameter wide partial circle.

–wpcircle100 100-pixel diameter wide partial circle.

–fcircle1 1-pixel diameter filled circle.

–fcircle10 10-pixel diameter filled circle.

–fcircle100 100-pixel diameter filled circle.

–fcircle500 500-pixel diameter filled circle.

–fcpcircle10 10-pixel diameter partial filled circle, chord fill, orientation and arc angle evenly distributed.

–fcpcircle100 100-pixel diameter partial filled circle, chord fill.

–fspcircle10 10-pixel diameter partial filled circle, pie slice fill, orientation and arc angle evenly distributed.

–fspcircle100 100-pixel diameter partial filled circle, pie slice fill.

–ellipse10 10-pixel diameter thin (line width 0) ellipse, major and minor axis sizes evenly distributed.

–ellipse100 100-pixel diameter thin ellipse.

–ellipse500 500-pixel diameter thin ellipse.

–dellipse100 100-pixel diameter thin dashed ellipse (3 on, 2 off).

–ddellipse100 100-pixel diameter thin double-dashed ellipse (3 fg, 2 bg).

–wellipse10 10-pixel diameter ellipse, line width 1.

–wellipse100 100-pixel diameter ellipse, line width 10.

–wellipse500 500-pixel diameter ellipse, line width 50.

–wdellipse100 100-pixel diameter dashed ellipse, line width 10 (30 on, 20 off).

–wddellipse100 100-pixel diameter double-dashed ellipse, line width 10 (30 fg, 20 bg).

–pellipse10 10-pixel diameter thin partial ellipse.

–pellipse100 100-pixel diameter thin partial ellipse.

–wpellipse10 10-pixel diameter wide partial ellipse.

–wpellipse100 100-pixel diameter wide partial ellipse.

–fellipse10 10-pixel diameter filled ellipse.

–fellipse100 100-pixel diameter filled ellipse.

–fellipse500 500-pixel diameter filled ellipse.

–fcpellipse10 10-pixel diameter partial filled ellipse, chord fill.

–fcpellipse100 100-pixel diameter partial filled ellipse, chord fill.

–fspellipse10 10-pixel diameter partial filled ellipse, pie slice fill.

–fspellipse100 100-pixel diameter partial filled ellipse, pie slice fill.

–triangle1 Fill 1-pixel/side triangle.

–triangle10 Fill 10-pixel/side triangle.

–triangle100 Fill 100-pixel/side triangle.

–trap10 Fill 10x10 trapezoid.

–trap100 Fill 100x100 trapezoid.

–strap10 Fill 10x10 transparent stippled trapezoid, 8x8 stipple pattern.

–strap100 Fill 100x100 transparent stippled trapezoid, 8x8 stipple pattern.

–ostrap10 Fill 10x10 opaque stippled trapezoid, 8x8 stipple pattern.

–ostrap100 Fill 100x100 opaque stippled trapezoid, 8x8 stipple pattern.

–tiletrap10 Fill 10x10 tiled trapezoid, 4x4 tile pattern.

–tiletrap100 Fill 100x100 tiled trapezoid, 4x4 tile pattern.

–bigstrap10 Fill 10x10 transparent stippled trapezoid, 161x145 stipple pattern.

–bigstrap100 Fill 100x100 transparent stippled trapezoid, 161x145 stipple pattern.

–bigostrap10 Fill 10x10 opaque stippled trapezoid, 161x145 stipple pattern.

–bigostrap100 Fill 100x100 opaque stippled trapezoid, 161x145 stipple pattern.

–bigtiletrap10 Fill 10x10 tiled trapezoid, 161x145 tile pattern.

–bigtiletrap100 Fill 100x100 tiled trapezoid, 161x145 tile pattern.

–eschertiletrap10
Fill 10x10 tiled trapezoid, 216x208 tile pattern.

–eschertiletrap100
Fill 100x100 tiled trapezoid, 216x208 tile pattern.

–complex10 Fill 10-pixel/side complex polygon.

–complex100 Fill 100-pixel/side complex polygon.

–ftext Character in 80-char line (6x13).

–f8text Character in 70-char line (8x13).

–f9text Character in 60-char line (9x15).

–f14text16 2-byte character in 40-char line (k14).

–tr10text Character in 80-char line (Times-Roman 10).

–tr24text Character in 30-char line (Times-Roman 24).

–polytext Character in 20/40/20 line (6x13, Times-Roman 10, 6x13).

–fitext Character in 80-char image line (6x13).

–f8itext Character in 70-char image line (8x13).

–f9itext Character in 60-char image line (9x15).

–f14itext16 2-byte character in 40-char image line (k14).

–tr10itext Character in 80-char image line (Times-Roman 10).

–tr24itext Character in 30-char image line (Times-Roman 24).

–scroll10 Scroll 10x10 pixels vertically.

–scroll100 Scroll 100x100 pixels vertically.

–scroll500 Scroll 500x500 pixels vertically.

–copywinwin10 Copy 10x10 square from window to window.

–copywinwin100
Copy 100x100 square from window to window.

–copywinwin500
Copy 500x500 square from window to window.

–copypixwin10 Copy 10x10 square from pixmap to window.

–copypixwin100
Copy 100x100 square from pixmap to window.

–copypixwin500
Copy 500x500 square from pixmap to window.

–copywinpix10 Copy 10x10 square from window to pixmap.

–copywinpix100
Copy 100x100 square from window to pixmap.

–copywinpix500
Copy 500x500 square from window to pixmap.

–copypixpix10 Copy 10x10 square from pixmap to pixmap.

–copypixpix100 Copy 100x100 square from pixmap to pixmap.

–copypixpix500 Copy 500x500 square from pixmap to pixmap.

–copyplane10 Copy 10x10 1-bit deep plane.

–copyplane100 Copy 100x100 1-bit deep plane.

–copyplane500 Copy 500x500 1-bit deep plane.

–putimage10 PutImage 10x10 square.

–putimage100 PutImage 100x100 square.

–putimage500 PutImage 500x500 square.

–shmput10 PutImage 10x10 square, MIT shared memory extension.

–shmput100 PutImage 100x100 square, MIT shared memory extension.

–shmput500 PutImage 500x500 square, MIT shared memory extension.

–getimage10 GetImage 10x10 square.

–getimage100 GetImage 100x100 square.

–getimage500 GetImage 500x500 square.

–noop X protocol NoOperation.

–atom GetAtomName.

–prop GetProperty.

–gc Change graphics context.

–create Create child window and map using MapSubwindows.

–ucreate Create unmapped window.

–map Map child window via MapWindow on parent.

–unmap Unmap child window via UnmapWindow on parent.

–destroy Destroy child window via DestroyWindow parent.

–popup Hide/expose window via Map/Unmap popup window.

–move Move window.

–umove Moved unmapped window.

–movetree Move window via MoveWindow on parent.

–resize Resize window.

–uresize Resize unmapped window.

–circulate Circulate lowest window to top.

–ucirculate Circulate unmapped window to top.

X DEFAULTS

There are no X defaults used by this program.

SEE ALSO

X(1), xbench(1), x11perfcomp(1)

COPYRIGHT

AUTHORS

Joel McCormack
Phil Karlton
Susan Angebranndt
Chris Kent

NAME

x11perfcomp – X11 server performance comparison program

SYNTAX

x11perfcomp [–r | –ro] [–l label_file] files

DESCRIPTION

The *x11perfcomp* program merges the output of several *x11perf(1)* runs into a nice tabular format. It takes the results in each file, fills in any missing test results if necessary, and for each test shows the objects/second rate of each server. If invoked with the -r or -ro options, it shows the relative performance of each server to the first server.

Normally, *x11perfcomp* uses the first file specified to determine which specific tests it should report on. Some (non-DEC :) servers may fail to perform all tests. In this case, *x11perfcomp* automatically substitutes in a rate of 0.0 objects/second. Since the first file determines which tests to report on, this file must contain a superset of the tests reported in the other files, else *x11perfcomp* will fail.

You can provide an explicit list of tests to report on by using the -l switch to specify a file of labels. You can create a label file by using the -label option in *x11perf.*

OPTIONS

x11perfcomp accepts the options listed below:

–r Specifies that the output should also include relative server performance.

–ro Specifies that the output should include only relative server performance.

–l label_file Specifies a label file to use.

X DEFAULTS

There are no X defaults used by this program.

SEE ALSO

X(1), x11perf(1)

AUTHORS

Mark Moraes wrote the original scripts to compare servers.
Joel McCormack just munged them together a bit.

NAME

xauth - X authority file utility

SYNOPSIS

xauth [-f *authfile*] [-vqib] [*command arg...*]

DESCRIPTION

The *xauth* program is used to edit and display the authorization information used in connecting to the X server. This program is usually to extract authorization records from one machine and merge them in on another (as is the case when using remote logins or to grant access to other users). Commands (described below) may be entered interactively, on the *xauth* command line, or in scripts. Note that this program does **not** contact the X server.

OPTIONS

The following options may be used with *xauth*. They may be given individually (e.g. *–q –i*) or may combined (e.g. *-qi*):

–f *authfile*
This option specifies the name of the authority file to use. By default, *xauth* will use the file specified by the XAUTHORITY environment variable or *.Xauthority* in the user's home directory.

–q This option indicates that *xauth* should operate quietly and not print unsolicited status messages. This is the default if an *xauth* command is is given on the command line or if the standard output is not directed to a terminal.

–v This option indicates that *xauth* should operate verbosely and print status messages indicating the results of various operations (e.g. how many records have been read in or written out). This is the default if *xauth* is reading commands from its standard input and its standard output is directed to a terminal.

–i This option indicates that *xauth* should ignore any authority file locks. Normally, *xauth* will refuse to read or edit any authority files that have been locked by other programs (usually *xdm* or another *xauth*).

–b This option indicates that *xauth* should attempt to break any authority file locks before proceeding and should only be used to clean up stale locks.

COMMANDS

The following commands may be used to manipulate authority files:

add *displayname protocolname hexkey*
An authorization entry for the indicated display using the given protocol and key data is added to the authorization file. The data is specified as an even-lengthed string of hexadecimal digits, each pair representing one octet. The first digit of each pair gives the most significant 4 bits of the octet and the second digit of the pairgives the least significant 4 bits. For example, a 32 character hexkey would represent a 128-bit value. A protocol name consisting of just a single period is treated as an abbreviation for *MIT-MAGIC-COOKIE-1*.

[n]extract *filename displayname...*
Authorization entries for each of the specified displays are written to the indicated file. If the *nextract* command is used, the entries are written in a numeric format suitable for non-binary transmission (such as secure electronic mail). The extracted entries can be read back in using the *merge* and *nmerge* commands. If the the filename consists of just a single dash, the entries will be written to the standard output.

[n]list [*displayname...*]
Authorization entries for each of the specified displays (or all if no displays are named) are printed on the standard output. If the *nlist* command is used, entries will be shown in the numeric format used by the *nextract* command; otherwise, they are shown in a textual format. Key data is always displayed in the hexadecimal format given in the description of the *add* command.

[n]merge [*filename*...]
Authorization entries are read from the specified files and are merged into the authorization database, superceding any matching existing entries. If the *nmerge* command is used, the numeric format given in the description of the *extract* command is used. If a filename consists of just a single dash, the standard input will be read if it hasn't been read before.

remove *displayname*...
Authorization entries matching the specified displays are removed from the authority file.

source *filename*
The specified file is treated as a script containing *xauth* commands to execute. Blank lines and lines beginning with a sharp sign (#) are ignored. A single dash may be used to indicate the standard input, if it hasn't already been read.

info Information describing the authorization file, whether or not any changes have been made, and from where *xauth* commands are being read is printed on the standard output.

exit If any modifications have been made, the authority file is written out (if allowed), and the program exits. An end of file is treated as an implicit *exit* command.

quit The program exits, ignoring any modifications. This may also be accomplished by pressing the interrupt character.

help [*string***]**
A description of all commands that begin with the given string (or all commands if no string is given) is printed on the standard output.

? A short list of the valid commands is printed on the standard output.

DISPLAY NAMES

Display names for the *add*, *[n]extract*, *[n]list*, *[n]merge*, and *remove* commands use the same format as the DISPLAY environment variable and the common *-display* command line argument. Display-specific information (such as the screen number) is unnecessary and will be ignored. Same-machine connections (such as local-host sockets, shared memory, and the Internet Protocol hostname *localhost*) are referred to as *hostname*/unix:*displaynumber* so that local entries for different machines may be stored in one authority file.

EXAMPLE

The most common use for *xauth* is to extract the entry for the current display, copy it to another machine, and merge it into the user's authority file on the remote machine:

```
% xauth extract - $DISPLAY | rsh other xauth merge -
```

ENVIRONMENT

This *xauth* program uses the following environment variables:

XAUTHORITY
to get the name of the authority file to use if the *–f* option isn't used. If this variable is not set, *xauth* will use *.Xauthority* in the user's home directory.

HOME to get the user's home directory if XAUTHORITY isn't defined.

BUGS

Users that have unsecure networks should take care to use encrypted file transfer mechanisms to copy authorization entries between machines. Similarly, the *MIT-MAGIC-COOKIE-1* protocol is not very useful in unsecure environments. Sites that are interested in additional security may need to use encrypted authorization mechanisms such as Kerberos.

Spaces are currently not allowed in the protocol name. Quoting could be added for the truly perverse.

COPYRIGHT

AUTHOR

Jim Fulton, MIT X Consortium

NAME

xbiff - mailbox flag for X

SYNOPSIS

xbiff [-*toolkitoption* ...] [-option ...]

DESCRIPTION

The *xbiff* program displays a little image of a mailbox. When there is no mail, the flag on the mailbox is down. When mail arrives, the flag goes up and the mailbox beeps. By default, pressing any mouse button in the image forces *xbiff* to remember the current size of the mail file as being the ''empty'' size and to lower the flag.

This program is nothing more than a wrapper around the Athena *Mailbox* widget.

OPTIONS

Xbiff accepts all of the standard X Toolkit command line options along with the additional options listed below:

–help This option indicates that a brief summary of the allowed options should be printed on the standard error.

–update *seconds*
This option specifies the frequency in seconds at which *xbiff* should update its display. If the mailbox is obscured and then exposed, it will be updated immediately. The default is 30 seconds.

–file *filename*
This option specifies the name of the file which should be monitored. By default, it watches /usr/spool/mail/*username*, where *username* is your login name.

–volume *percentage*
This option specifies how loud the bell should be rung when new mail comes in.

–shape This option indicates that the mailbox window should be shaped if masks for the empty or full images are given.

The following standard X Toolkit command line arguments are commonly used with *xbiff:*

–display *display*
This option specifies the X server to contact.

–geometry *geometry*
This option specifies the preferred size and position of the mailbox window. The mailbox is 48 pixels wide and 48 pixels high and will be centered in the window.

–bg *color*
This option specifies the color to use for the background of the window.

–bd *color*
This option specifies the color to use for the border of the window.

–bw *number*
This option specifies the width in pixels of the border surrounding the window.

–fg *color* This option specifies the color to use for the foreground of the window.

–rv This option indicates that reverse video should be simulated by swapping the foreground and background colors.

–xrm *resourcestring*
This option specifies a resource string to be used. This is especially useful for setting resources that do not have separate command line options.

X DEFAULTS

The application class name is XBiff. This program uses the *Mailbox* widget in the Athena widget set. It understands all of the core resource names and classes as well as:

checkCommand (class **CheckCommand)**
Specifies a shell command to be executed to check for new mail rather than examining the size of **file**. The specified string value is used as the argument to a *system*(3) call and may therefore contain i/o redirection. An exit status of 0 indicates that new mail is waiting, 1 indicates that there has been no change in size, and 2 indicates that the mail has been cleared. By default, no shell command is provided.

file (class **File)**
Specifies the name of the file to monitor. The default is to watch /usr/spool/mail/*username*, where *username* is your login name.

onceOnly (class **Boolean)**
Specifies that the bell is only rung the first time new mail is found and is not rung again until at least one interval has passed with no mail waiting. The window will continue to indicate the presence of new mail until it has been retrieved. The default is false.

width (class **Width)**
Specifies the width of the mailbox.

height (class **Height)**
Specifies the height of the mailbox.

update (class **Interval)**
Specifies the frequency in seconds at which the mail should be checked. The default is 30.

volume (class **Volume)**
Specifies how loud the bell should be rung. The default is 33 percent.

foreground (class **Foreground)**
Specifies the color for the foreground.

reverseVideo (class **ReverseVideo)**
Specifies that the foreground and background should be reversed.

flip (class **Flip)**
Specifies whether or not the image that is shown when mail has arrived should be inverted. The default is ‘‘true.’’

fullPixmap (class **Pixmap)**
Specifies a bitmap to be shown when new mail has arrived. The default is flagup.

emptyPixmap (class **Pixmap)**
Specifies a bitmap to be shown when no new mail is present. The default is flagdown.

shapeWindow (class **ShapeWindow)**
Specifies whether or not the mailbox window should be shaped to the given fullPixmapMask and emptyPixmapMask. The default is false.

fullPixmapMask (class **PixmapMask)**
Specifies a mask for the bitmap to be shown when new mail has arrived. The default is none.

emptyPixmapMask (class **PixmapMask)**
Specifies a mask for the bitmap to be shown when no new mail is present. The default is none.

ACTIONS

The *Mailbox* widget provides the following actions for use in event translations:

check() This action causes the widget to check for new mail and display the flag appropriately.

unset() This action causes the widget to lower the flag until new mail comes in.

set() This action causes the widget to raise the flag until the user resets it.

The default translation is

<ButtonPress>: unset()

ENVIRONMENT

DISPLAY
to get the default host and display number.

XENVIRONMENT
to get the name of a resource file that overrides the global resources stored in the RESOURCE_MANAGER property.

SEE ALSO

X(1), xrdb(1), stat(2)

BUGS

The mailbox bitmaps are ugly.

COPYRIGHT

AUTHOR

Jim Fulton, MIT X Consortium
Additional hacks by Ralph Swick, DEC/MIT Project Athena

NAME

xcalc – scientific calculator for X

SYNOPSIS

xcalc [-stipple] [-rpn] [-*toolkitoption...*]

DESCRIPTION

xcalc is a scientific calculator desktop accessory that can emulate a TI-30 or an HP-10C.

OPTIONS

xcalc accepts all of the standard toolkit command line options along with two additional options:

–stipple This option indicates that the background of the calculator should be drawn using a stipple of the foreground and background colors. On monochrome displays improves the appearance.

–rpn This option indicates that Reverse Polish Notation should be used. In this mode the calculator will look and behave like an HP-10C. Without this flag, it will emulate a TI-30.

OPERATION

Pointer Usage: Operations may be performed with pointer button 1, or in some cases, with the keyboard. Many common calculator operations have keyboard accelerators. To quit, press pointer button 3 on the AC key of the TI calculator, or the ON key of the HP calculator.

Calculator Key Usage (TI mode): The numbered keys, the +/- key, and the +, -, ∗, /, and = keys all do exactly what you would expect them to. It should be noted that the operators obey the standard rules of precedence. Thus, entering "3+4∗5=" results in "23", not "35". The parentheses can be used to override this. For example, "(1+2+3)∗(4+5+6)=" results in "6∗15=90".

The entire number in the calculator display can be selected, in order to paste the result of a calculation into text.

The action procedures associated with each function are given below. These are useful if you are interested in defining a custom calculator. The action used for all digit keys is **digit(***n***)**, where *n* is the corresponding digit, 0..9.

1/x Replaces the number in the display with its reciprocal. The corresponding action procedure is **reciprocal()**.

xˆ2 Squares the number in the display. The corresponding action procedure is **square()**.

SQRT Takes the square root of the number in the display. The corresponding action procedure is **squareRoot()**.

CE/C When pressed once, clears the number in the display without clearing the state of the machine. Allows you to re-enter a number if you make a mistake. Pressing it twice clears the state, also. The corresponding action procedure for TI mode is **clear()**.

AC Clears the display, the state, and the memory. Pressing it with the third pointer button turns off the calculator, in that it exits the program. The action procedure to clear the state is **off()**; to quit, **quit()**.

INV Invert function. See the individual function keys for details. The corresponding action procedure is **inverse()**.

sin Computes the sine of the number in the display, as interpreted by the current DRG mode (see DRG, below). If inverted, it computes the arcsine. The corresponding action procedure is **sine()**.

cos Computes the cosine, or arccosine when inverted. The corresponding action procedure is **cosine()**.

tan Computes the tangent, or arctangent when inverted. The corresponding action procedure is **tangent()**.

DRG Changes the DRG mode, as indicated by ’DEG’, ’RAD’, or ’GRAD’ at the bottom of of the calculator ‘‘liquid crystal’’ display. When in ’DEG’ mode, numbers in the display are taken as being degrees. In ’RAD’ mode, numbers are in radians, and in ’GRAD’ mode, numbers are in grads. When inverted, the DRG key has a feature of converting degrees to radians to grads and vice-versa. Example: put the calculator into ’DEG’ mode, and enter "45 INV DRG". The display should now show something along the lines of ".785398", which is 45 degrees converted to radians. The corresponding action procedure is **degree()**.

e The constant ’e’. (2.7182818...). The corresponding action procedure is **e()**.

EE Used for entering exponential numbers. For example, to get "-2.3E-4" you’d enter "2 . 3 +/- EE 4 +/-". The corresponding action procedure is **scientific()**.

log Calculates the log (base 10) of the number in the display. When inverted, it raises "10.0" to the number in the display. For example, entering "3 INV log" should result in "1000". The corresponding action procedure is **logarithm()**.

ln Calculates the log (base e) of the number in the display. When inverted, it raises "e" to the number in the display. For example, entering "e ln" should result in "1". The corresponding action procedure is **naturalLog()**.

yˆx Raises the number on the left to the power of the number on the right. For example "2 yˆx 3 =" results in "8", which is 2ˆ3. For a further example, "(1+2+3) yˆx (1+2) =" equals "6 yˆx 3" which equals "216". The corresponding action procedure is **power()**.

PI The constant ’pi’. (3.1415927....) The corresponding action procedure is **pi()**.

x! Computes the factorial of the number in the display. The number in the display must be an integer in the range 0-500, though, depending on your math library, it might overflow long before that. The corresponding action procedure is **factorial()**.

(Left parenthesis. The corresponding action procedure for TI calculators is **leftParen()**.

) Right parenthesis. The corresponding action procedure for TI calculators is **rightParen()**.

/ Division. The corresponding action procedure is **divide()**.

∗ Multiplication. The corresponding action procedure is **multiply()**.

- Subtraction. The corresponding action procedure is **subtract()**.

+ Addition. The corresponding action procedure is **add()**.

= Perform calculation. The TI-specific action procedure is **equal()**.

STO Copies the number in the display to the memory location. The corresponding action procedure is **store()**.

RCL Copies the number from the memory location to the display. The corresponding action procedure is **recall()**.

SUM Adds the number in the display to the number in the memory location. The corresponding action procedure is **sum()**.

EXC Swaps the number in the display with the number in the memory location. The corresponding action procedure for the TI calculator is **exchange()**.

+/- Negate; change sign. The corresponding action procedure is **negate()**.

. Decimal point. The action procedure is **decimal()**.

Calculator Key Usage (RPN mode): The number keys, CHS (change sign), +, -, ∗, /, and ENTR keys all do exactly what you would expect them to do. Many of the remaining keys are the same as in TI mode. The differences are detailed below. The action procedure for the ENTR key is **enter()**.

<- This is a backspace key that can be used if you make a mistake while entering a number. It will erase digits from the display. (See BUGS). Inverse backspace will clear the X register. The corresponding action procedure is **back()**.

ON Clears the display, the state, and the memory. Pressing it with the third pointer button turns off the calculator, in that it exits the program. To clear state, the action procedure is **off**; to quit, **quit()**.

INV Inverts the meaning of the function keys. This would be the *f* key on an HP calculator, but *xcalc* does not display multiple legends on each key. See the individual function keys for details.

10ˆx Raises "10.0" to the number in the top of the stack. When inverted, it calculates the log (base 10) of the number in the display. The corresponding action procedure is **tenpower()**.

eˆx Raises "e" to the number in the top of the stack. When inverted, it calculates the log (base e) of the number in the display. The action procedure is **epower()**.

STO Copies the number in the top of the stack to a memory location. There are 10 memory locations. The desired memory is specified by following this key with a digit key.

RCL Pushes the number from the specified memory location onto the stack.

SUM Adds the number on top of the stack to the number in the specified memory location.

x:y Exchanges the numbers in the top two stack positions, the X and Y registers. The corresponding action procedure is **XexchangeY()**.

R v Rolls the stack downward. When inverted, it rolls the stack upward. The corresponding action procedure is **roll()**.

blank These keys were used for programming functions on the HP-10C. Their functionality has not been duplicated in *xcalc*.

Finally, there are two additional action procedures: **bell()**, which rings the bell; and **selection()**, which performs a cut on the entire number in the calculator's ‘‘liquid crystal’’ display.

ACCELERATORS

Accelerators are shortcuts for entering commands. *xcalc* provides some sample keyboard accelerators; also users can customize accelerators. The numeric keypad accelerators provided by *xcalc* should be intuitively correct. The accelerators defined by *xcalc* on the main keyboard are given below:

TI Key	HP Key	Keyboard Accelerator	TI Function	HP Function
SQRT	SQRT	r	squareRoot()	squareRoot()
AC	ON	space	clear()	clear()
AC	<-	Delete	clear()	back()
AC	<-	Backspace	clear()	back()
AC	<-	Control-H	clear()	back()
AC		Clear	clear()	
AC	ON	q	quit()	quit()
AC	ON	Control-C	quit()	quit()
INV	i	i	inverse()	inverse()
sin	s	s	sine()	sine()
cos	c	c	cosine()	cosine()
tan	t	t	tangent()	tangent()
DRG	DRG	d	degree()	degree()
e		e	e()	
ln	ln	l	naturalLog()	naturalLog()
yˆx	yˆx	ˆ	power()	power()

PI	PI	p	pi()	pi()
x!	x!	!	factorial()	factorial()
(		(	leftParen()	
)		)	rightParen()	
/	/	/	divide()	divide()
*	*	*	multiply()	multiply()
-	-	-	subtract()	subtract()
+	+	+	add()	add()
=		=	equal()	
0..9	0..9	0..9	digit()	digit()
.	.	.	decimal()	decimal()
+/-	CHS	n	negate()	negate()
	x:y	x		XexchangeY()
	ENTR	Return		enter()
	ENTR	Linefeed		enter()

CUSTOMIZATION

The application class name is XCalc.

xcalc has an enormous application defaults file which specifies the position, label, and function of each key on the calculator. It also gives translations to serve as keyboard accelerators. Because these resources are not specified in the source code, you can create a customized calculator by writing a private application defaults file, using the Athena Command and Form widget resources to specify the size and position of buttons, the label for each button, and the function of each button.

The foreground and background colors of each calculator key can be individually specified. For the TI calculator, a classical color resource specification might be:

```
XCalc.ti.Command.background:    gray50
XCalc.ti.Command.foreground:    white
```

For each of buttons 20, 25, 30, 35, and 40, specify:

```
XCalc.ti.button20.background:    black
XCalc.ti.button20.foreground:    white
```

For each of buttons 22, 23, 24, 27, 28, 29, 32, 33, 34, 37, 38, and 39:

```
XCalc.ti.button22.background:    white
XCalc.ti.button22.foreground:    black
```

WIDGET HIERARCHY

In order to specify resources, it is useful to know the hierarchy of the widgets which compose *xcalc*. In the notation below, indentation indicates hierarchical structure. The widget class name is given first, followed by the widget instance name.

```
XCalc xcalc
        Form ti or hp    (the name depends on the mode)
                Form bevel
                        Form screen
                                Label M
                                Toggle LCD
                                Label INV
                                Label DEG
                                Label RAD
```

Label GRAD
Label P
Command button1
Command button2
Command button3

and so on, ...

Command button38
Command button39
Command button40

APPLICATION RESOURCES

rpn (Class **Rpn**)
Specifies that the rpn mode should be used. The default is TI mode.

stipple (Class **Stipple**)
Indicates that the background should be stippled. The default is ''on'' for monochrome displays, and ''off'' for color displays.

cursor (Class **Cursor**)
The name of the symbol used to represent the pointer. The default is ''hand2''.

COLORS

If you would like xcalc to use its ti colors, include the following in the #ifdef COLOR section of the file you read with xrdb:

```
*customization:        -color
```

This will cause xcalc to pick up the colors in the app-defaults color customization file: /usr/lib/X11/app-defaults/XCalc-color.

SEE ALSO

X(1), xrdb(1), the Athena Widget Set

BUGS

HP mode: A bug report claims that the sequence of keys 5, ENTER, <- should clear the display, but it doesn't.

COPYRIGHT

AUTHORS

John Bradley, University of Pennsylvania
Mark Rosenstein, MIT Project Athena
Donna Converse, MIT X Consortium

NAME

xclipboard - X clipboard client

SYNOPSIS

xclipboard [*-toolkitoption* ...] [-w] [-nw]

DESCRIPTION

The *xclipboard* program is used to collect and display text selections that are sent to the CLIPBOARD by other clients. It is typically used to save CLIPBOARD selections for later use. It stores each CLIPBOARD selection as a separate string, each of which can be selected. Each time CLIPBOARD is asserted by another application, *xclipboard* transfers the contents of that selection to a new buffer and displays it in the text window. Buffers are never automatically deleted, so you'll want to use the delete button to get rid of useless items.

Since *xclipboard* uses a Text Widget to display the contents of the clipboard, text sent to the CLIPBOARD may be re-selected for use in other applications. *xclipboard* also responds to requests for the CLIPBOARD selection from other clients by sending the entire contents of the currently displayed buffer.

An *xclipboard* window has the following buttons across the top:

quit When this button is pressed, *xclipboard* exits.

delete When this button is pressed, the current buffer is deleted and the next one displayed.

new Creates a new buffer with no contents. Useful in constructing a new CLIPBOARD selection by hand.

next Displays the next buffer in the list.

previous Displays the previous buffer.

OPTIONS

The *xclipboard* program accepts all of the standard X Toolkit command line options as well as the following:

–w This option indicates that lines of text that are too long to be displayed on one line in the clipboard should wrap around to the following lines.

–nw This option indicates that long lines of text should not wrap around. This is the default behavior.

WIDGETS

In order to specify resources, it is useful to know the hierarchy of the widgets which compose *xclipboard*. In the notation below, indentation indicates hierarchical structure. The widget class name is given first, followed by the widget instance name.

```
XClipboard  xclipboard
        Form  form
                Command  quit
                Command  delete
                Command  new
                Command  next
                Command  prev
                Text  text
```

SENDING/RETRIEVING CLIPBOARD CONTENTS

Text is copied to the clipboard whenever a client asserts ownership of the **CLIPBOARD** selection. Text is copied from the clipboard whenever a client requests the contents of the **CLIPBOARD** selection. Examples of event bindings that a user may wish to include in a resource configuration file to use the clipboard are:

```
*VT100.Translations: #override \
```

<Btn3Up>:	select-end(CLIPBOARD) \n\
<Btn2Up>:	insert-selection(PRIMARY,CLIPBOARD) \n\
<Btn2Down>:	ignore ()

SEE ALSO

X(1), xcutsel(1), xterm(1), individual client documentation for how to make a selection and send it to the CLIPBOARD.

ENVIRONMENT

DISPLAY

to get the default host and display number.

XENVIRONMENT

to get the name of a resource file that overrides the global resources stored in the RESOURCE_MANAGER property.

FILES

/usr/lib/X11/app-defaults/XClipboard - specifies required resources

COPYRIGHT

AUTHOR

Ralph R. Swick, DEC/MIT Project Athena
Chris D. Peterson, MIT X Consortium
Keith Packard, MIT X Consortium

NAME

xclock - analog / digital clock for X

SYNOPSIS

xclock [-*toolkitoption* ...] [-help] [-analog] [-digital] [-chime] [-hd *color*] [-hl *color*] [-update *seconds*] [-padding *number*]

DESCRIPTION

The *xclock* program displays the time in analog or digital form. The time is continuously updated at a frequency which may be specified by the user. This program is nothing more than a wrapper around the Athena Clock widget.

OPTIONS

Xclock accepts all of the standard X Toolkit command line options along with the additional options listed below:

–help This option indicates that a brief summary of the allowed options should be printed on the standard error.

–analog This option indicates that a conventional 12 hour clock face with tick marks and hands should be used. This is the default.

–digital This option indicates that a 24 hour digital clock should be used.

–chime This option indicates that the clock should chime once on the half hour and twice on the hour.

–hd *color*
This option specifies the color of the hands on an analog clock. The default is *black*.

–hl *color*
This option specifies the color of the edges of the hands on an analog clock, and is only useful on color displays. The default is *black*.

–update *seconds*
This option specifies the frequency in seconds at which *xclock* should update its display. If the clock is obscured and then exposed, it will be updated immediately. A value of less than 30 seconds will enable a second hand on an analog clock. The default is 60 seconds.

–padding *number*
This option specifies the width in pixels of the padding between the window border and clock text or picture. The default is 10 on a digital clock and 8 on an analog clock.

X DEFAULTS

This program uses the *Athena Clock* widget. It understands all of the core resource names and classes as well as:

width (class **Width)**
Specifies the width of the clock. The default for analog clocks is 164 pixels; the default for digital clocks is whatever is needed to hold the clock when displayed in the chosen font.

height (class **Height)**
Specifies the height of the clock. The default for analog clocks is 164 pixels; the default for digital clocks is whatever is needed to hold the clock when displayed in the chosen font.

update (class **Interval)**
Specifies the frequency in seconds at which the time should be redisplayed.

foreground (class **Foreground)**
Specifies the color for the tic marks. The default is depends on whether *reverseVideo* is specified. If *reverseVideo* is specified the default is *lwhite*, otherwise the default is *black*.

hands (class **Foreground)**
Specifies the color of the insides of the clock's hands. The default is depends on whether

reverseVideo is specified. If *reverseVideo* is specified the default is *lwhite*, otherwise the default is *black*.

highlight (class **Foreground)**
Specifies the color used to highlight the clock's hands. The default is
depends on whether *reverseVideo* is specified. If *reverseVideo* is specified the default is *lwhite*, otherwise the default is *black*.

analog (class **Boolean)**
Specifies whether or not an analog clock should be used instead of a digital one. The default is True.

chime (class **Boolean)**
Specifies whether or not a bell should be rung on the hour and half hour.

padding (class **Margin)**
Specifies the amount of internal padding in pixels to be used. The default is 8.

font (class **Font)**
Specifies the font to be used for the digital clock. Note that variable width fonts currently will not always display correctly.

WIDGETS

In order to specify resources, it is useful to know the hierarchy of the widgets which compose *xclock*. In the notation below, indentation indicates hierarchical structure. The widget class name is given first, followed by the widget instance name.

```
XClock  xclock
        Clock  clock
```

ENVIRONMENT

DISPLAY
to get the default host and display number.

XENVIRONMENT
to get the name of a resource file that overrides the global resources stored in the RESOURCE_MANAGER property.

FILES

/usr/lib/X11/app-defaults/XClock - specifies required resources

SEE ALSO

X(1), xrdb(1), time(3C), Athena Clock widget

BUGS

Xclock believes the system clock.

When in digital mode, the string should be centered automatically.

COPYRIGHT

AUTHORS

Tony Della Fera (MIT-Athena, DEC)
Dave Mankins (MIT-Athena, BBN)
Ed Moy (UC Berkeley)

NAME

xcmsdb – Xlib Screen Color Characterization Data utility

SYNOPSIS

xcmsdb [–option ...] [*filename*]

DESCRIPTION

xcmsdb is used to load, query, or remove Screen Color Characterization Data stored in properties on the root window of the screen. Screen Color Characterization Data is an integral part of Xlib, necessary for proper conversion between device-independent and device-dependent color specifications. Xlib uses the XDCCC_LINEAR_RGB_MATRICES and XDCCC_LINEAR_RGB_CORRECTION properties to store color characterization data for color monitors. It uses XDCCC_GRAY_SCREENWWHITEPOINT and XDCCC_GRAY_CORRECTION properties for gray scale monitors. Because Xlib allows the addition of Screen Color Characterization Function Sets, added function sets may place their Screen Color Characterization Data on other properties. This utility is unaware of these other properties, therefore, you will need to use a similar utility provided with the function set, or use the *xprop* utility.

The ASCII readable contents of *filename* (or the standard input if no input file is given) are appropriately transformed for storage in properties, provided the **–query** or **–remove** options are not specified.

OPTIONS

xcmsdb program accepts the following options:

–query This option attempts to read the XDCCC properties off the screen's root window. If successful, it transforms the data into a more readable format, then sends the data to standard out.

–remove This option attempts to remove the XDCCC properties on the screen's root window.

–color This option sets the query and remove options to only check for the XDCCC_LINEAR_RGB_MATRICES and XDCCC_LINEAR_RGB_CORRECTION properties. If the **–color** option is not set then the query and remove options check for all the properties.

–format 32 | 16 | 8
Specifies the property format (32, 16, or 8 bits per entry) for the XDCCC_LINEAR_RGB_CORRECTION property. Precision of encoded floating point values increases with the increase in bits per entry. The default is 32 bits per entry.

SEE ALSO

xprop(1), Xlib documentation

ENVIRONMENT

DISPLAY
to figure out which display and screen to use.

BUGS

Unknown

COPYRIGHT

AUTHOR

Chuck Adams, Tektronix Inc.

NAME

xcmstest – XCMS test program

SYNOPSIS

xcmstest [-display displayname]

or

xcmstest -echo [-display displayname] < cmdfile

DESCRIPTION

xcmstest was developed as a simple test interface for testing the XCMS API library during its development.

BEFORE INVOCATION

Insure that the Screen Color Characterization Data has been placed in the appropriate properties on the screen's root window. To load these properties, refer to the xcmsdb utility provided with this release.

In order to convert color names using a client-side color name database, the database file must be specified via the XCMSDB environment variable.

To use the sample database provided with this release set XCMSDB as follows:
% setenv XCMSDB ‘pwd‘/Xcms.txt

USING XCMSTEST

Upon invocation of xcmstest, you should receive a "XCMS >" prompt. At this point you can begin entering commands.

To list the commands currently available enter "list":
XCMS > list

To receive help for a specific command enter the command name followed by "-help". For example:
XCMS > LookupColor -help

The -echo option provides a more readable output, echoing the command from the cmdfile. At some point in time, you might try using some simple test cases provided with the release (in the tc directory, e.g. test1, test2, AllocNColor, gamutquery). For example, to use commmand file test1, invoke xcmstest as follows:
% xcmstest -echo < tc/test1

CAVEATS

Color name strings passed to xcmstest commands cannot contain a space. However, since spaces are ignored by the XCMS API library, you can pass a color name with the spaces eliminated. For example:
XCMS > LookupColor cornflowerblue UNDEFINED RGB

SEE ALSO

xcmsdb(1), Chapter 6 of Xlib - C Language X Interface.

COPYRIGHT

NAME

xconsole - monitor system console messages

SYNOPSIS

xconsole [-*toolkitoption* ...] [-file *file-name*] [-notify] [-stripNonprint] [-daemon] [-verbose] [-exitOnFail]

DESCRIPTION

The *xconsole* program displays messages which are usually sent to /dev/console.

OPTIONS

Xconsole accepts all of the standard X Toolkit command line options along with the additional options listed below:

–file *file-name*

To monitor some other device, use this option to specify the device name. This does not work on regular files as they are always ready to be read from.

–notify –nonotify

When new data are received from the console and the notify option is set, the icon name of the application has " *" appended, so that it is evident even when the application is iconified. –notify is the default.

–daemon

This option causes *xconsole* to place itself in the background, using fork/exit.

–verbose

When set, this option directs *xconsole* to display an informative message in the first line of the text buffer.

–exitOnFail

When set, this option directs *xconsole* to exit when it is unable to redirect the console output.

X DEFAULTS

This program uses the *Athena Text* widget, look in the *Athena Widget Set* documentation for controlling it.

WIDGETS

In order to specify resources, it is useful to know the hierarchy of the widgets which compose *xconsole*. In the notation below, indentation indicates hierarchical structure. The widget class name is given first, followed by the widget instance name.

```
XConsole  xconsole
        XConsole  text
```

ENVIRONMENT

DISPLAY

to get the default host and display number.

XENVIRONMENT

to get the name of a resource file that overrides the global resources stored in the RESOURCE_MANAGER property.

FILES

/usr/lib/X11/app-defaults/XConsole - specifies required resources

SEE ALSO

X(1), xrdb(1), Athena Text widget

COPYRIGHT

AUTHOR

Keith Packard (MIT X Consortium)

NAME

xcutsel - interchange between cut buffer and selection

SYNOPSIS

xcutsel [-*toolkitoption* ...] [-selection *selection*] [-cutbuffer *number*]

DESCRIPTION

The *xcutsel* program is used to copy the current selection into a cut buffer and to make a selection that contains the current contents of the cut buffer. It acts as a bridge between applications that don't support selections and those that do.

By default, *xcutsel* will use the selection named PRIMARY and the cut buffer CUT_BUFFER0. Either or both of these can be overridden by command line arguments or by resources.

An *xcutsel* window has the following buttons:

quit When this button is pressed, *xcutsel* exits. Any selections held by *xcutsel* are automatically released.

copy PRIMARY to 0
When this button is pressed, *xcutsel* copies the current selection into the cut buffer.

copy 0 to PRIMARY
When this button is pressed, *xcutsel* converts the current contents of the cut buffer into the selection.

The button labels reflect the selection and cutbuffer selected by command line options or through the resource database.

When the ''copy 0 to PRIMARY'' button is activated, the button will remain inverted as long as *xcutsel* remains the owner of the selection. This serves to remind you which client owns the current selection. Note that the value of the selection remains constant; if the cutbuffer is changed, you must again activate the copy button to retrieve the new value when desired.

OPTIONS

Xcutsel accepts all of the standard X Toolkit command line options as well as the following:

–selection *name*
This option specifies the name of the selection to use. The default is PRIMARY. The only supported abbreviations for this option are ''-select'', ''-sel'' and ''-s'', as the standard toolkit option ''-selectionTimeout'' has a similar name.

–cutbuffer *number*
This option specifies the cut buffer to use. The default is cut buffer 0.

X DEFAULTS

This program accepts all of the standard X Toolkit resource names and classes as well as:

selection (class **Selection)**
This resource specifies the name of the selection to use. The default is PRIMARY.

cutBuffer (class **CutBuffer)**
This resource specifies the number of the cut buffer to use. The default is 0.

WIDGET NAMES

The following instance names may be used when user configuration of the labels in them is desired:

sel-cut (class **Command)**
This is the ''copy SELECTION to BUFFER'' button.

cut-sel (class **Command)**
This is the ''copy BUFFER to SELECTION'' button.

quit (class **Command)**
This is the ''quit'' button.

SEE ALSO

X(1), xclipboard(1), xterm(1), text widget documentation, individual client documentation for how to make a selection.

BUGS

There is no way to change the name of the selection or the number of the cut buffer while the program is running.

COPYRIGHT

AUTHOR

Ralph R. Swick, DEC/MIT Project Athena

NAME

xditview – display ditroff output

SYNOPSIS

xditview [–*toolkitoption* . . .] [–*option* . . .] [*filename*]

DESCRIPTION

The *xditview* program displays *ditroff* output on an X display. It uses no special metrics and automatically converts the printer coordinates into screen coordinates; using the user-specified screen resolution, rather than the actual resolution so that the appropriate fonts can be found. If ''**–**'' is given as the *filename, xditview* reads from standard input. If ''|'' is the first character of *filename, xditview* forks *sh* to run the rest of the ''file name'' and uses the standard output of that command.

OPTIONS

Xditview accepts all of the standard X Toolkit command line options along with the additional options listed below:

–page *page-number*
: This option specifies the page number of the document to be displayed at start up time.

–resolution *screen-resolution*
: This specifies the desired screen resolution to use; fonts will be opened by requesting this resolution field in the XLFD names.

–noPolyText
: Some X servers incorrectly implement PolyText with multiple strings per request. This option suppesses the use of this feature in *xditview.*

–backingStore *backing-store-type*
: Redisplay can take up to a second or so; this option causes the server to save the window contents so that when it is scrolled around the viewport, the window is painted from contents saved in backing store. *backing-store-type* can be one of **Always**, **WhenMapped** or **NotUseful**.

The following standard X Toolkit command line arguments are commonly used with *xditview:*

–bg *color*
: This option specifies the color to use for the background of the window. The default is *white*.

–bd *color*
: This option specifies the color to use for the border of the window. The default is *black*.

–bw *number*
: This option specifies the width in pixels of the border surrounding the window.

–fg *color*
: This option specifies the color to use for displaying text. The default is *black*.

–fn *font*
: This option specifies the font to be used for displaying widget text. The default is *fixed*.

–rv
: This option indicates that reverse video should be simulated by swapping the foreground and background colors.

–geometry *geometry*
: This option specifies the preferred size and position of the window.

–display *host***:***display*
: This option specifies the X server to contact.

–xrm *resourcestring*
: This option specifies a resource string to be used.

X DEFAULTS

This program uses a *Dvi* widget. It understands all of the core resource names and classes as well as:

width (class **Width)**
Specifies the width of the window.

height (class **Height)**
Specifies the height of the window.

foreground (class **Foreground)**
Specifies the default foreground color.

font (class **Font)**
Specifies the font to be used for error messages.

FontMap (class FontMap)
To associate the *ditroff* fonts with appropriate X fonts, this string resource contains a set of newline separated specifications, each of which consists of a ditroff name, some white space and an XLFD pattern with ∗ characters in appropriate places to allow all sizes to be listed. The default fontMap is:

```
R    –∗–times–medium–r–normal––∗–∗–∗–∗–∗–∗–iso8859–1\n\
I    –∗–times–medium–i–normal––∗–∗–∗–∗–∗–∗–iso8859–1\n\
B    –∗–times–bold–r–normal––∗–∗–∗–∗–∗–∗–iso8859–1\n\
F    –∗–times–bold–i–normal––∗–∗–∗–∗–∗–∗–iso8859–1\n\
BI   –∗–times–bold–i–normal––∗–∗–∗–∗–∗–∗–iso8859–1\n\
C    –∗–courier–medium–r–normal––∗–∗–∗–∗–∗–∗–iso8859–1\n\
CO   –∗–courier–medium–o–normal––∗–∗–∗–∗–∗–∗–iso8859–1\n\
CB   –∗–courier–bold–r–normal––∗–∗–∗–∗–∗–∗–iso8859–1\n\
CF   –∗–courier–bold–o–normal––∗–∗–∗–∗–∗–∗–iso8859–1\n\
H    –∗–helvetica–medium–r–normal––∗–∗–∗–∗–∗–∗–iso8859–1\n\
HO   –∗–helvetica–medium–o–normal––∗–∗–∗–∗–∗–∗–iso8859–1\n\
HB   –∗–helvetica–bold–r–normal––∗–∗–∗–∗–∗–∗–iso8859–1\n\
HF   –∗–helvetica–bold–o–normal––∗–∗–∗–∗–∗–∗–iso8859–1\n\
N    –∗–new century schoolbook–medium–r–normal––∗–∗–∗–∗–∗–∗–iso8859–1\n\
NI   –∗–new century schoolbook–medium–i–normal––∗–∗–∗–∗–∗–∗–iso8859–1\n\
NB   –∗–new century schoolbook–bold–r–normal––∗–∗–∗–∗–∗–∗–iso8859–1\n\
NF   –∗–new century schoolbook–bold–i–normal––∗–∗–∗–∗–∗–∗–iso8859–1\n\
A    –∗–charter–medium–r–normal––∗–∗–∗–∗–∗–∗–iso8859–1\n\
AI   –∗–charter–medium–i–normal––∗–∗–∗–∗–∗–∗–iso8859–1\n\
AB   –∗–charter–bold–r–normal––∗–∗–∗–∗–∗–∗–iso8859–1\n\
AF   –∗–charter–bold–i–normal––∗–∗–∗–∗–∗–∗–iso8859–1\n\
S    –∗–symbol–medium–r–normal––∗–∗–∗–∗–∗–∗–adobe–fontspecific\n\
S2   –∗–symbol–medium–r–normal––∗–∗–∗–∗–∗–∗–adobe–fontspecific\n
```

USING XDITVIEW WITH DITROFF

You can use any ditroff output file with *xditview,* although files which use the fonts appropriate to the fontMap will look more accurate on the screen. On servers which support scaled fonts, all requested font sizes will be accurately reflected on the screen; for servers which do not support scaled *xditview* will use the closest font from the same family.

SEE ALSO

X(1), *xrdb*(1), *ditroff*(1), *X Logical Font Description Conventions*

ORIGIN

Portions of this program originated in *xtroff which was derived* from *suntroff.*

COPYRIGHT

AUTHORS

Keith Packard (MIT X Consortium)
Richard L. Hyde (Purdue)
David Slattengren (Berkeley)
Malcom Slaney (Schlumberger Palo Alto Research)
Mark Moraes (University of Toronto)

NAME

xdm – X Display Manager with support for XDMCP

SYNOPSIS

xdm [**–config** *configuration_file*] [**–nodaemon**] [**–debug** *debug_level*] [**–error** *error_log_file*] [**–resources** *resource_file*] [**–server** *server_entry*] [**–session** *session_program*]

DESCRIPTION

Xdm manages a collection of X displays, which may be on the local host or remote servers. The design of *xdm* was guided by the needs of X terminals as well as the X Consortium standard XDMCP, the *X Display Manager Control Protocol*. *Xdm* provides services similar to those provided by *init*, *getty* and *login* on character terminals: prompting for login name and password, authenticating the user, and running a ‘‘session.’’

A ‘‘session’’ is defined by the lifetime of a particular process; in the traditional character-based terminal world, it is the user’s login shell. In the *xdm* context, it is an arbitrary session manager. This is because in a windowing environment, a user’s login shell process does not necessarily have any terminal-like interface with which to connect. When a real session manager is not available, a window manager or terminal emulator is typically used as the ‘‘session manager,’’ meaning that termination of this process terminates the user’s session.

When the session is terminated, *xdm* resets the X server and (optionally) restarts the whole process.

When *xdm* receives an Indirect query via XDMCP, it can run a *chooser* process to perform an XDMCP BroadcastQuery (or an XDMCP Query to specified hosts) on behalf of the display and offer a menu of possible hosts that offer XDMCP display management. This feature is useful with X terminals that do not offer a host menu themselves.

Because *xdm* provides the first interface that users will see, it is designed to be simple to use and easy to customize to the needs of a particular site. *Xdm* has many options, most of which have reasonable defaults. Browse through the various sections of this manual, picking and choosing the things you want to change. Pay particular attention to the **Session Program** section, which will describe how to set up the style of session desired.

TYPICAL USAGE

Actually, *xdm* is designed to operate in such a wide variety of environments that *typical* is probably a misnomer.

First, the *xdm* configuration file should be set up. Make a directory (usually */usr/lib/X11/xdm*) to contain all of the relevant files. Here is a reasonable configuration file, which could be named *xdm-config*:

```
DisplayManager.servers:          /usr/lib/X11/xdm/Xservers
DisplayManager.errorLogFile:     /usr/lib/X11/xdm/xdm-errors
DisplayManager*resources:        /usr/lib/X11/xdm/Xresources
DisplayManager*startup:          /usr/lib/X11/xdm/Xstartup
DisplayManager*session:          /usr/lib/X11/xdm/Xsession
DisplayManager.pidFile:          /usr/lib/X11/xdm/xdm-pid
DisplayManager._0.authorize:     true
DisplayManager*authorize:        false
```

Note that this file simply contains references to other files. Note also that some of the resources are specified with ‘‘*’’ separating the components. These resources can be made unique for each different display, by replacing the ‘‘*’’ with the display-name, but normally this is not very useful. See the **Resources** section for a complete discussion.

The first file, */usr/lib/X11/xdm/Xservers,* contains the list of displays to manage that are not using XDMCP. Most workstations have only one display, numbered 0, so the file will look something like this:

```
:0 Local local /usr/bin/X11/X :0
```

This will keep */usr/bin/X11/X* running on this display and manage a continuous cycle of sessions.

The file */usr/lib/X11/xdm/xdm-errors* will contain error messages from *xdm* and anything output to stderr by *Xsetup*, *Xstartup*, *Xsession* or *Xreset*. When you have trouble getting *xdm* working, check this file to see if *xdm* has any clues to the trouble.

The next configuration entry, */usr/lib/X11/xdm/Xresources*, is loaded onto the display as a resource database using *xrdb*. As the authentication widget reads this database before starting up, it usually contains parameters for that widget:

```
xlogin*login.translations: #override\
        <Key>F1: set-session-argument(failsafe) finish-field()\n\
        <Key>Return: set-session-argument() finish-field()
xlogin*borderWidth: 3
#ifdef COLOR
xlogin*greetColor: CadetBlue
xlogin*failColor: red
#endif
```

Please note the translations entry; it specifies a few new translations for the widget which allow users to escape from the default session (and avoid troubles that may occur in it). Note that if #override is not specified, the default translations are removed and replaced by the new value, not a very useful result as some of the default translations are quite useful (such as ‘‘<Key>: insert-char ()’’ which responds to normal typing).

The *Xstartup* file shown here simply prevents login while the file */etc/nologin* exists. As there is no provision for displaying any messages here (there isn’t any core X client which displays files), the user will probably be baffled by this behavior. Thus this is not a complete example, but simply a demonstration of the available functionality.

Here is a sample *Xstartup* script:

```
#!/bin/sh
#
# Xstartup
#
# This program is run as root after the user is verified
#
if [ –f /etc/nologin ]; then
        exit 1
fi
exit 0
```

The most interesting script is *Xsession*. This version recognizes the special ‘‘failsafe’’ mode, specified in the translations in the *Xresources* file above, to provide an escape from the ordinary session:

```
#!/bin/sh
#
# Xsession
#
# This is the program that is run as the client
# for the display manager. This example is
# quite friendly as it attempts to run a per-user
# .xsession file instead of forcing a particular
```

```
# session layout
#

case $# in
1)
        case $1 in
        failsafe)
                exec xterm –geometry 80x24–0–0 –ls
                ;;
        esac
esac

startup=$HOME/.xsession
resources=$HOME/.Xresources

if [ –f $startup ]; then
        exec $startup
        exec /bin/sh $startup
else
        if [ ! –f $resources ]; then
                resources=$HOME/.Xdefaults
        fi
        if [ –f $resources ]; then
                xrdb –load $resources
        fi
        twm &
        exec xterm –geometry 80x24+10+10 –ls
fi
```

OPTIONS

All of these options, except **–config**, specify values that can also be specified in the configuration file as resources.

–config *configuration_file*
: Names the configuration file, which specifies resources to control the behavior of *xdm*. */usr/lib/X11/xdm/xdm-config* is the default.

–nodaemon
: Specifies ''false'' as the value for the **DisplayManager.daemonMode** resource. This suppresses the normal daemon behavior, which is for *xdm* to close all file descriptors, disassociate itself from the controlling terminal, and put itself in the background when it first starts up.

–debug *debug_level*
: Specifies the numeric value for the **DisplayManager.debugLevel** resource. A non-zero value causes *xdm* to print lots of debugging statements to the terminal; it also disables the **DisplayManager.daemonMode** resource, forcing *xdm* to run synchronously. To interpret these debugging messages, a copy of the source code for *xdm* is almost a necessity. No attempt has been made to rationalize or standardize the output.

–error *error_log_file*
: Specifies the value for the **DisplayManager.errorLogFile** resource. This file contains errors from *xdm* as well as anything written to stderr by the various scripts and programs run during the progress of the session.

–resources *resource_file*
: Specifies the value for the **DisplayManager*resources** resource. This file is loaded using *xrdb* to

specify configuration parameters for the authentication widget.

–server *server_entry*
: Specifies the value for the **DisplayManager.servers** resource. See the section **Server Specification** for a description of this resource.

–udpPort *port_number*
: Specifies the value for the **DisplayManager.requestPort** resource. This sets the port-number which *xdm* will monitor for XDMCP requests. As XDMCP uses the registered well-known UDP port 177, this resource should not be changed except for debugging.

–session *session_program*
: Specifies the value for the **DisplayManager*session** resource. This indicates the program to run as the session after the user has logged in.

–xrm *resource_specification*
: Allows an arbitrary resource to be specified, as in most X Toolkit applications.

RESOURCES

At many stages the actions of *xdm* can be controlled through the use of its configuration file, which is in the X resource format. Some resources modify the behavior of *xdm* on all displays, while others modify its behavior on a single display. Where actions relate to a specific display, the display name is inserted into the resource name between ‘‘DisplayManager’’ and the final resource name segment. For example, **DisplayManager.expo_0.startup** is the name of the resource which defines the startup shell file on the ‘‘expo:0’’ display. Because the resource manager uses colons to separate the name of the resource from its value and dots to separate resource name parts, *xdm* substitutes underscores for both dots and colons when generating the resource name.

DisplayManager.servers
: This resource either specifies a file name full of server entries, one per line (if the value starts with a slash), or a single server entry. See the section **Server Specification** for the details.

DisplayManager.requestPort
: This indicates the UDP port number which *xdm* uses to listen for incoming XDMCP requests. Unless you need to debug the system, leave this with its default value of 177.

DisplayManager.errorLogFile
: Error output is normally directed at the system console. To redirect it, set this resource to a file name. A method to send these messages to *syslog* should be developed for systems which support it; however, the wide variety of interfaces precludes any system-independent implementation. This file also contains any output directed to stderr by the *Xsetup, Xstartup, Xsession* and *Xreset* files, so it will contain descriptions of problems in those scripts as well.

DisplayManager.debugLevel
: If the integer value of this resource is greater than zero, reams of debugging information will be printed. It also disables daemon mode, which would redirect the information into the bit-bucket, and allows non-root users to run *xdm,* which would normally not be useful.

DisplayManager.daemonMode
: Normally, *xdm* attempts to make itself into a daemon process unassociated with any terminal. This is accomplished by forking and leaving the parent process to exit, then closing file descriptors and releasing the controlling terminal. In some environments this is not desired (in particular, when debugging). Setting this resource to ‘‘false’’ will disable this feature.

DisplayManager.pidFile
: The filename specified will be created to contain an ASCII representation of the process-id of the main *xdm* process. *Xdm* also uses file locking on this file to attempt to eliminate multiple daemons running on the same machine, which would cause quite a bit of havoc.

DisplayManager.lockPidFile
: This is the resource which controls whether *xdm* uses file locking to keep multiple display

managers from running amok. On System V, this uses the *lockf* library call, while on BSD it uses *flock.*

DisplayManager.authDir

This names a directory in which *xdm* stores authorization files while initializing the session. The default value is */usr/lib/X11/xdm.*

DisplayManager.autoRescan

This boolean controls whether *xdm* rescans the configuration, servers, access control and authentication keys files after a session terminates and the files have changed. By default it is ''true.'' You can force *xdm* to reread these files by sending a SIGHUP to the main process.

DisplayManager.removeDomainname

When computing the display name for XDMCP clients, the name resolver will typically create a fully qualified host name for the terminal. As this is sometimes confusing, *xdm* will remove the domain name portion of the host name if it is the same as the domain name of the local host when this variable is set. By default the value is ''true.''

DisplayManager.keyFile

XDM-AUTHENTICATION-1 style XDMCP authentication requires that a private key be shared between *xdm* and the terminal. This resource specifies the file containing those values. Each entry in the file consists of a display name and the shared key. By default, *xdm* does not include support for XDM-AUTHENTICATION-1, as it requires DES which is not generally distributable because of United States export restrictions.

DisplayManager.accessFile

To prevent unauthorized XDMCP service and to allow forwarding of XDMCP IndirectQuery requests, this file contains a database of hostnames which are either allowed direct access to this machine, or have a list of hosts to which queries should be forwarded to. The format of this file is described in the section **XDMCP Access Control.**

DisplayManager.exportList

A whitespace-separated list of additional environment variables to pass on to the *Xsetup*, *Xstartup*, *Xsession*, and *Xreset* programs.

DisplayManager.randomFile

A file to checksum to generate the seed of authorization keys. This should be a file that changes frequently. The default is */dev/mem*.

DisplayManager.DISPLAY.resources

This resource specifies the name of the file to be loaded by *xrdb* as the resource database onto the root window of screen 0 of the display. The *Xsetup* program, the Login widget, and *chooser* will use the resources set in this file. This resource data base is loaded just before the authentication procedure is started, so it can control the appearance of the login window. See the section **Authentication Widget,** which describes the various resources that are appropriate to place in this file. There is no default value for this resource, but */usr/lib/X11/xdm/Xresources* is the conventional name.

DisplayManager.DISPLAY.chooser

Specifies the program run to offer a host menu for Indirect queries redirected to the special host name CHOOSER. */usr/lib/X11/xdm/chooser* is the default. See the sections **XDMCP Access Control** and **Chooser**.

DisplayManager.DISPLAY.xrdb

Specifies the program used to load the resources. By default, *xdm* uses */usr/bin/X11/xrdb*.

DisplayManager.DISPLAY.cpp

This specifies the name of the C preprocessor which is used by *xrdb*.

DisplayManager.DISPLAY.setup

This specifies a program which is run (as root) before offering the Login window. This may be used to change the appearence of the screen around the Login window or to put up other windows (e.g., you may want to run *xconsole* here). By default, no program is run. The conventional name for a file used here is *Xsetup*. See the section **Setup Program.**

DisplayManager.DISPLAY.startup
This specifies a program which is run (as root) after the authentication process succeeds. By default, no program is run. The conventional name for a file used here is *Xstartup*. See the section **Startup Program.**

DisplayManager.DISPLAY.session
This specifies the session to be executed (not running as root). By default, */usr/bin/X11/xterm* is run. The conventional name is *Xsession*. See the section **Session Program.**

DisplayManager.DISPLAY.reset
This specifies a program which is run (as root) after the session terminates. Again, by default no program is run. The conventional name is *Xreset*. See the section **Reset Program.**

DisplayManager.DISPLAY.openDelay

DisplayManager.DISPLAY.openRepeat

DisplayManager.DISPLAY.openTimeout

DisplayManager.DISPLAY.startAttempts
These numeric resources control the behavior of *xdm* when attempting to open intransigent servers. **openDelay** is the length of the pause (in seconds) between successive attempts, **openRepeat** is the number of attempts to make, **openTimeout** is the amount of time to wait while actually attempting the open (i.e., the maximum time spent in the *connect*(2) system call) and **startAttempts** is the number of times this entire process is done before giving up on the server. After **openRepeat** attempts have been made, or if **openTimeout** seconds elapse in any particular attempt, *xdm* terminates and restarts the server, attempting to connect again. This process is repeated **startAttempts** times, at which point the display is declared dead and disabled. Although this behavior may seem arbitrary, it has been empirically developed and works quite well on most systems. The default values are 5 for **openDelay**, 5 for **openRepeat**, 30 for **openTimeout** and 4 for **startAttempts**.

DisplayManager.DISPLAY.pingInterval

DisplayManager.DISPLAY.pingTimeout
To discover when remote displays disappear, *xdm* occasionally pings them, using an X connection and *XSync* calls. **pingInterval** specifies the time (in minutes) between each ping attempt, **pingTimeout** specifies the maximum amount of time (in minutes) to wait for the terminal to respond to the request. If the terminal does not respond, the session is declared dead and terminated. By default, both are set to 5 minutes. If you frequently use X terminals which can become isolated from the managing host, you may wish to increase this value. The only worry is that sessions will continue to exist after the terminal has been accidentally disabled. *xdm* will not ping local displays. Although it would seem harmless, it is unpleasant when the workstation session is terminated as a result of the server hanging for NFS service and not responding to the ping.

DisplayManager.DISPLAY.terminateServer
This boolean resource specifies whether the X server should be terminated when a session terminates (instead of resetting it). This option can be used when the server tends to grow without bound over time, in order to limit the amount of time the server is run. The default value is ‘‘false.’’

DisplayManager.DISPLAY.userPath
Xdm sets the PATH environment variable for the session to this value. It should be a colon separated list of directories; see *sh*(1) for a full description. ‘‘:/bin:/usr/bin:/usr/bin/X11:/usr/ucb’’ is a common setting. The default value can be specified at build time in the X system configuration file with DefaultUserPath;

DisplayManager.DISPLAY.systemPath

Xdm sets the PATH environment variable for the startup and reset scripts to the value of this resource. The default for this resource is specified at build time by the DefaultSystemPath entry in the system configuration file; ‘‘/etc:/bin:/usr/bin:/usr/bin/X11:/usr/ucb’’ is a common choice. Note the absence of ‘‘.’’ from this entry. This is a good practice to follow for root; it avoids many common Trojan Horse system penetration schemes.

DisplayManager.DISPLAY.systemShell

Xdm sets the SHELL environment variable for the startup and reset scripts to the value of this resource. It is */bin/sh* by default.

DisplayManager.DISPLAY.failsafeClient

If the default session fails to execute, *xdm* will fall back to this program. This program is executed with no arguments, but executes using the same environment variables as the session would have had (see the section **Session Program**). By default, */usr/bin/X11/xterm* is used.

DisplayManager.DISPLAY.grabServer

DisplayManager.DISPLAY.grabTimeout

To improve security, *xdm* grabs the server and keyboard while reading the login name and password. The **grabServer** resource specifies if the server should be held for the duration of the name/password reading. When ‘‘false,’’ the server is ungrabbed after the keyboard grab succeeds, otherwise the server is grabbed until just before the session begins. The default is ‘‘false.’’ The **grabTimeout** resource specifies the maximum time *xdm* will wait for the grab to succeed. The grab may fail if some other client has the server grabbed, or possibly if the network latencies are very high. This resource has a default value of 3 seconds; you should be cautious when raising it, as a user can be spoofed by a look-alike window on the display. If the grab fails, *xdm* kills and restarts the server (if possible) and the session.

DisplayManager.DISPLAY.authorize

DisplayManager.DISPLAY.authName

authorize is a boolean resource which controls whether *xdm* generates and uses authorization for the local server connections. If authorization is used, **authName** is a whitespace-separated list of authorization mechanisms to use. XDMCP connections dynamically specify which authorization mechanisms are supported, so **authName** is ignored in this case. When **authorize** is set for a display and authorization is not available, the user is informed by having a different message displayed in the login widget. By default, **authorize** is ‘‘true’’; **authName** is ‘‘MIT-MAGIC-COOKIE-1.’’

DisplayManager.DISPLAY.authFile

This file is used to communicate the authorization data from *xdm* to the server, using the **–auth** server command line option. It should be kept in a directory which is not world-writable as it could easily be removed, disabling the authorization mechanism in the server.

DisplayManager.DISPLAY.authComplain

If set to ‘‘false,’’ disables the use of the **unsecureGreeting** in the login window. See the section **Authentication Widget.** The default is ‘‘true.’’

DisplayManager.DISPLAY.resetSignal

The number of the signal *xdm* sends to reset the server. See the section **Controlling the Server.** The default is 1 (SIGHUP).

DisplayManager.DISPLAY.termSignal

The number of the signal *xdm* sends to terminate the server. See the section **Controlling the Server.** The default is 15 (SIGTERM).

DisplayManager.DISPLAY.resetForAuth

The original implementation of authorization in the sample server reread the authorization file at server reset time, instead of when checking the initial connection. As *xdm* generates the authorization information just before connecting to the display, an old server would not get up-to-date authorization information. This resource causes *xdm* to send SIGHUP to the server after setting up the file, causing an additional server reset to occur, during which time the new authorization information will be read. The default is ‘‘false,’’ which will work for all MIT servers.

DisplayManager.DISPLAY.userAuthDir

When *xdm* is unable to write to the usual user authorization file ($HOME/.Xauthority), it creates a unique file name in this directory and points the environment variable XAUTHORITY at the created file. It uses */tmp* by default.

XDMCP ACCESS CONTROL

The database file specified by the **DisplayManager.accessFile** provides information which *xdm* uses to control access from displays requesting XDMCP service. This file contains three types of entries: entries which control the response to Direct and Broadcast queries, entries which control the response to Indirect queries, and macro definitions.

The format of the Direct entries is simple, either a host name or a pattern, which is distinguished from a host name by the inclusion of one or more meta characters (‘∗’ matches any sequence of 0 or more characters, and ‘?’ matches any single character) which are compared against the host name of the display device. If the entry is a host name, all comparisons are done using network addresses, so any name which converts to the correct network address may be used. For patterns, only canonical host names are used in the comparison, so ensure that you do not attempt to match aliases. Preceding either a host name or a pattern with a ‘!’ character causes hosts which match that entry to be excluded.

An Indirect entry also contains a host name or pattern, but follows it with a list of host names or macros to which indirect queries should be sent.

A macro definition contains a macro name and a list of host names and other macros that the macro expands to. To distinguish macros from hostnames, macro names start with a ‘%’ character. Macros may be nested.

Indirect entries may also specify to have *xdm* run *chooser* to offer a menu of hosts to connect to. See the section **Chooser**.

When checking access for a particular display host, each entry is scanned in turn and the first matching entry determines the response. Direct and Broadcast entries are ignored when scanning for an Indirect entry and vice-versa.

Blank lines are ignored, ‘#’ is treated as a comment delimiter causing the rest of that line to be ignored, and ‘*newline*’ causes the newline to be ignored, allowing indirect host lists to span multiple lines.

Here is an example Xaccess file:

```
#
# Xaccess – XDMCP access control file
#

#
# Direct/Broadcast query entries
#

!xtra.lcs.mit.edu          # disallow direct/broadcast service for xtra
bambi.ogi.edu              # allow access from this particular display
∗.lcs.mit.edu              # allow access from any display in LCS

#
# Indirect query entries
#
```

%HOSTS	expo.lcs.mit.edu xenon.lcs.mit.edu \ excess.lcs.mit.edu kanga.lcs.mit.edu	
extract.lcs.mit.edu	xenon.lcs.mit.edu	#force extract to contact xenon
!xtra.lcs.mit.edu	dummy	#disallow indirect access
∗.lcs.mit.edu	%HOSTS	#all others get to choose

CHOOSER

For X terminals that do not offer a host menu for use with Broadcast or Indirect queries, the *chooser* program can do this for them. In the *Xaccess* file, specify ‘‘CHOOSER’’ as the first entry in the Indirect host list. *Chooser* will send a Query request to each of the remaining host names in the list and offer a menu of all the hosts that respond.

The list may consist of the word ‘‘BROADCAST,’’ in which case *chooser* will send a Broadcast instead, again offering a menu of all hosts that respond. Note that on some operating systems, UDP packets cannot be broadcast, so this feature will not work.

Example *Xaccess* file using *chooser*:

extract.lcs.mit.edu	CHOOSER %HOSTS	#offer a menu of these hosts
xtra.lcs.mit.edu	CHOOSER BROADCAST	#offer a menu of all hosts

The program to use for *chooser* is specified by the **DisplayManager.DISPLAY.chooser** resource. Resources for this program can be put into the file named by **DisplayManager.DISPLAY.resources**.

SERVER SPECIFICATION

The resource **DisplayManager.servers** gives a server specification or, if the values starts with a slash (/), the name of a file containing server specifications, one per line.

Each specification indicates a display which should constantly be managed and which is not using XDMCP. Each consists of at least three parts: a display name, a display class, a display type, and (for local servers) a command line to start the server. A typical entry for local display number 0 would be:

```
:0 Digital-QV local /usr/bin/X11/X :0
```

The display types are:

local	local display: *xdm* must run the server
foreign	remote display: *xdm* opens an X connection to a running server

The display name must be something that can be passed in the **–display** option to an X program. This string is used to generate the display-specific resource names, so be careful to match the names (e.g. use ‘‘:0 local /usr/bin/X11/X :0’’ instead of ‘‘localhost:0 local /usr/bin/X11/X :0’’ if your other resources are specified as ‘‘DisplayManager._0.session’’). The display class portion is also used in the display-specific resources, as the class of the resource. This is useful if you have a large collection of similar displays (like a corral of X terminals) and would like to set resources for groups of them. When using XDMCP, the display is required to specify the display class, so the manual for your particular X terminal should document the display class string for your device. If it doesn’t, you can run *xdm* in debug mode and look at the resource strings which it generates for that device, which will include the class string.

SETUP PROGRAM

The *Xsetup* file is run after the server is reset, but before the Login window is offered. The file is typically a shell script. It is run as root, so should be careful about security. This is the place to change the root background or bring up other windows that should appear on the screen along with the Login widget.

In addition to any specified by **DisplayManager.exportList**, the following environment variables are passed:

DISPLAY	the associated display name
PATH	the value of **DisplayManager.DISPLAY.systemPath**
SHELL	the value of **DisplayManager.DISPLAY.systemShell**
XAUTHORITY	may be set to an authority file

Note that since *xdm* grabs the keyboard, any other windows will not be able to receive keyboard input. They will be able to interact with the mouse, however; beware of potential security holes here. If **DisplayManager.DISPLAY.grabServer** is set, *Xsetup* will not be able to connect to the display at all. Resources for this program can be put into the file named by **DisplayManager.DISPLAY.resources**.

AUTHENTICATION WIDGET

The authentication widget reads a name/password pair from the keyboard. Nearly every imaginable parameter can be controlled with a resource. Resources for this widget should be put into the file named by **DisplayManager.DISPLAY.resources**. All of these have reasonable default values, so it is not necessary to specify any of them.

xlogin.Login.width, xlogin.Login.height, xlogin.Login.x, xlogin.Login.y
The geometry of the Login widget is normally computed automatically. If you wish to position it elsewhere, specify each of these resources.

xlogin.Login.foreground
The color used to display the typed-in user name.

xlogin.Login.font
The font used to display the typed-in user name.

xlogin.Login.greeting
A string which identifies this window. The default is ‘‘X Window System.’’

xlogin.Login.unsecureGreeting
When X authorization is requested in the configuration file for this display and none is in use, this greeting replaces the standard greeting. The default is ‘‘This is an unsecure session’’

xlogin.Login.greetFont
The font used to display the greeting.

xlogin.Login.greetColor
The color used to display the greeting.

xlogin.Login.namePrompt
The string displayed to prompt for a user name. *Xrdb* strips trailing white space from resource values, so to add spaces at the end of the prompt (usually a nice thing), add spaces escaped with backslashes. The default is ‘‘Login: ’’

xlogin.Login.passwdPrompt
The string displayed to prompt for a password. The default is ‘‘Password: ’’

xlogin.Login.promptFont
The font used to display both prompts.

xlogin.Login.promptColor
The color used to display both prompts.

xlogin.Login.fail
A message which is displayed when the authentication fails. The default is ‘‘Login incorrect’’

xlogin.Login.failFont
The font used to display the failure message.

xlogin.Login.failColor
The color used to display the failure message.

xlogin.Login.failTimeout
The number of seconds that the failure message is displayed. The default is 30.

xlogin.Login.translations

This specifies the translations used for the login widget. Refer to the X Toolkit documentation for a complete discussion on translations. The default translation table is:

```
Ctrl<Key>H:          delete-previous-character() \n\
Ctrl<Key>D:          delete-character() \n\
Ctrl<Key>B:          move-backward-character() \n\
Ctrl<Key>F:          move-forward-character() \n\
Ctrl<Key>A:          move-to-begining() \n\
Ctrl<Key>E:          move-to-end() \n\
Ctrl<Key>K:          erase-to-end-of-line() \n\
Ctrl<Key>U:          erase-line() \n\
Ctrl<Key>X:          erase-line() \n\
Ctrl<Key>C:          restart-session() \n\
Ctrl<Key>\\:         abort-session() \n\
<Key>BackSpace:      delete-previous-character() \n\
<Key>Delete:         delete-previous-character() \n\
<Key>Return:         finish-field() \n\
<Key>:               insert-char() \
```

The actions which are supported by the widget are:

delete-previous-character
: Erases the character before the cursor.

delete-character
: Erases the character after the cursor.

move-backward-character
: Moves the cursor backward.

move-forward-character
: Moves the cursor forward.

move-to-begining
: (Apologies about the spelling error.) Moves the cursor to the beginning of the editable text.

move-to-end
: Moves the cursor to the end of the editable text.

erase-to-end-of-line
: Erases all text after the cursor.

erase-line
: Erases the entire text.

finish-field
: If the cursor is in the name field, proceeds to the password field; if the cursor is in the password field, checks the current name/password pair. If the name/password pair is valid, *xdm* starts the session. Otherwise the failure message is displayed and the user is prompted again.

abort-session
: Terminates and restarts the server.

abort-display
: Terminates the server, disabling it. This is a rash action and is not accessible in the default configuration. It can be used to stop *xdm* when shutting the system down or when using *xdmshell*.

restart-session
: Resets the X server and starts a new session. This can be used when the resources have been

changed and you want to test them or when the screen has been overwritten with system messages.

insert-char
: Inserts the character typed.

set-session-argument
: Specifies a single word argument which is passed to the session at startup. See the sections **Session Program** and **Typical Usage**.

allow-all-access
: Disables access control in the server. This can be used when the .Xauthority file cannot be created by *xdm.* Be very careful using this; it might be better to disconnect the machine from the network before doing this.

STARTUP PROGRAM

The *Xstartup* file is typically a shell script. It is run as root and should be very careful about security. This is the place to put commands which add entries to */etc/utmp,* mount users' home directories from file servers, display the message of the day, or abort the session if logins are not allowed.

In addition to any specified by **DisplayManager.exportList**, the following environment variables are passed:

DISPLAY	the associated display name
HOME	the initial working directory of the user
USER	the user name
PATH	the value of **DisplayManager.DISPLAY.systemPath**
SHELL	the value of **DisplayManager.DISPLAY.systemShell**
XAUTHORITY	may be set to an authority file

No arguments are passed to the script. *Xdm* waits until this script exits before starting the user session. If the exit value of this script is non-zero, *xdm* discontinues the session and starts another authentication cycle.

SESSION PROGRAM

The *Xsession* program is the command which is run as the user's session. It is run with the permissions of the authorized user.

In addition to any specified by **DisplayManager.exportList**, the following environment variables are passed:

DISPLAY	the associated display name
HOME	the initial working directory of the user
USER	the user name
PATH	the value of **DisplayManager.DISPLAY.userPath**
SHELL	the user's default shell (from *getpwnam*)
XAUTHORITY	may be set to a non-standard authority file

At most installations, *Xsession* should look in $HOME for a file *.xsession,* which contains commands that each user would like to use as a session. *Xsession* should also implement a system default session if no user-specified session exists. See the section **Typical Usage**.

An argument may be passed to this program from the authentication widget using the 'set-session-argument' action. This can be used to select different styles of session. One good use of this feature is to allow the user to escape from the ordinary session when it fails. This allows users to repair their own *.xsession* if it fails, without requiring administrative intervention. The section **Typical Usage** demonstrates this feature.

RESET PROGRAM

Symmetrical with *Xstartup*, the *Xreset* script is run after the user session has terminated. Run as root, it should contain commands that undo the effects of commands in *Xstartup*, removing entries from */etc/utmp* or unmounting directories from file servers. The environment variables that were passed to *Xstartup* are also passed to *Xreset*.

CONTROLLING THE SERVER

Xdm controls local servers using POSIX signals. SIGHUP is expected to reset the server, closing all client connections and performing other cleanup duties. SIGTERM is expected to terminate the server. If these signals do not perform the expected actions, the resources **DisplayManager.DISPLAY.resetSignal** and **DisplayManager.DISPLAY.termSignal** can specify alternate signals.

To control remote terminals not using XDMCP, *xdm* searches the window hierarchy on the display and uses the protocol request KillClient in an attempt to clean up the terminal for the next session. This may not actually kill all of the clients, as only those which have created windows will be noticed. XDMCP provides a more sure mechanism; when *xdm* closes its initial connection, the session is over and the terminal is required to close all other connections.

CONTROLLING XDM

Xdm responds to two signals: SIGHUP and SIGTERM. When sent a SIGHUP, *xdm* rereads the configuration file, the access control file, and the servers file. For the servers file, it notices if entries have been added or removed. If a new entry has been added, *xdm* starts a session on the associated display. Entries which have been removed are disabled immediately, meaning that any session in progress will be terminated without notice and no new session will be started.

When sent a SIGTERM, *xdm* terminates all sessions in progress and exits. This can be used when shutting down the system.

Xdm attempts to mark its various sub-processes for *ps*(1) by editing the command line argument list in place. Because *xdm* can't allocate additional space for this task, it is useful to start *xdm* with a reasonably long command line (using the full path name should be enough). Each process which is servicing a display is marked **–***display.*

OTHER POSSIBILITIES

You can use *xdm* to run a single session at a time, using the 4.3 *init* options or other suitable daemon by specifying the server on the command line:

```
xdm –server ":0 SUN-3/60CG4 local /usr/bin/X :0"
```

Or, you might have a file server and a collection of X terminals. The configuration for this is identical to the sample above, except the *Xservers* file would look like

```
extol:0 VISUAL-19 foreign
exalt:0 NCD-19 foreign
explode:0 NCR-TOWERVIEW3000 foreign
```

This directs *xdm* to manage sessions on all three of these terminals. See the section **Controlling Xdm** for a description of using signals to enable and disable these terminals in a manner reminiscent of *init*(8).

LIMITATIONS

One thing that *xdm* isn't very good at doing is coexisting with other window systems. To use multiple window systems on the same hardware, you'll probably be more interested in *xinit*.

FILES

/usr/lib/X11/xdm/xdm-config
the default configuration file

/usr/lib/X11/xdm/Xaccess

the default access file, listing authorized displays

/usr/lib/X11/xdm/Xservers
the default server file, listing non-XDMCP servers to manage

$(HOME)/.Xauthority user authorization file where *xdm* stores keys for clients to read

/usr/lib/X11/xdm/chooser
the default chooser

/usr/bin/X11/xrdb the default resource database loader

/usr/bin/X11/X the default server

/usr/bin/X11/xterm the default session program and failsafe client

/usr/lib/X11/xdm/A<host>–<suffix>
the default place for authorization files

SEE ALSO

X(1), *xinit*(1), *xauth*(1), *Xsecurity*(1), and XDMCP

COPYRIGHT

AUTHOR

Keith Packard, MIT X Consortium

NAME

xdpr – dump an X window directly to a printer

SYNOPSIS

xdpr [*filename*] [**–display** *host:display*] [**–P***printer*] [**–device** *devtype*] [*option* . . .]

DESCRIPTION

Xdpr uses the commands *xwd*, *xpr*, and *lpr* or *lp* to dump an X window, process it for a particular printer type, and print it out on the printer of your choice. This is the easiest way to get a printout of a window. *Xdpr* by default will print the largest possible representation of the window on the output page.

The options for *xdpr* are the same as those for *xpr*, *xwd*, and *lpr* or *lp*. The most commonly-used options are described below; see the manual pages for these commands for more detailed descriptions of the many options available.

OPTIONS

filename
: Specifies a file containing a window dump (created by *xwd*) to be printed instead of selecting an X window.

–P*printer*
: Specifies a printer to send the output to. If a printer name is not specified here, *xdpr* (really, *lpr* or *lp*) will send your output to the printer specified by the *PRINTER* environment variable. Be sure that type of the printer matches the type specified with the *–device* option.

–display *host:display***[**.*screen***]**
: Normally, *xdpr* gets the host and display number to use from the environment variable ''DISPLAY.'' One can, however, specify them explicitly; see *X*(1).

–device *devtype*
: Specifies the device on which the file will be printed. Currently supported:

la100	Digital LA100
ljet	HP LaserJet series and other monochrome PCL devices such as ThinkJet, QuietJet, RuggedWriter, HP2560 series, and HP2930 series printers
ln03	Digital LN03
pjet	HP PaintJet (color mode)
pjetxl	HP HP PaintJet XL Color Graphics Printer (color mode)
pp	IBM PP3812
ps	PostScript printer

The default is PostScript. **–device lw** (LaserWriter) is equivalent to **–device ps** and is provided only for backwards compatibility.

–help This option displays the list of options known to *xdpr.*

Any other arguments will be passed to the *xwd, xpr,* and *lpr* or *lp* commands as appropriate for each.

SEE ALSO

xwd(1), *xpr*(1), *lpr*(1), *lp*(1), *xwud*(1), *X*(1)

ENVIRONMENT

DISPLAY which display to use by default.

PRINTER which printer to use by default.

COPYRIGHT

AUTHORS

Paul Boutin, MIT Project Athena

Michael R. Gretzinger, MIT Project Athena
Jim Gettys, MIT Project Athena

NAME

xdpyinfo – display information utility for X

SYNOPSIS

xdpyinfo [–display *displayname*]

DESCRIPTION

Xdpyinfo is a utility for displaying information about an X server. It is used to examine the capabilities of a server, the predefined values for various parameters used in communicating between clients and the server, and the different types of screens and visuals that are available.

EXAMPLE

The following shows a sample produced by *xdpyinfo* when connected to display that supports an 8 plane screen and a 1 plane screen.

```
name of display:    :0.0
version number:    11.0
vendor string:    MIT X Consortium
vendor release number:    4
maximum request size:  16384 longwords (65536 bytes)
motion buffer size:  0
bitmap unit, bit order, padding:    32, MSBFirst, 32
image byte order:    MSBFirst
number of supported pixmap formats:    2
supported pixmap formats:
    depth 1, bits_per_pixel 1, scanline_pad 32
    depth 8, bits_per_pixel 8, scanline_pad 32
keycode range:    minimum 8, maximum 129
focus:  PointerRoot
number of extensions:    4
    SHAPE
    MIT-SHM
    Multi-Buffering
    MIT-SUNDRY-NONSTANDARD
default screen number:    0
number of screens:    2

screen #0:
  dimensions:    1152x900 pixels (325x254 millimeters)
  resolution:    90x90 dots per inch
  depths (2):    1, 8
  root window id:    0x8006e
  depth of root window:    8 planes
  number of colormaps:    minimum 1, maximum 1
  default colormap:    0x8006b
  default number of colormap cells:    256
  preallocated pixels:    black 1, white 0
  options:    backing-store YES, save-unders YES
  current input event mask:    0xd0801d
    KeyPressMask             ButtonPressMask          ButtonReleaseMask
    EnterWindowMask          ExposureMask             SubstructureRedirectMask
    PropertyChangeMask       ColormapChangeMask
  number of visuals:    6
  default visual id:  0x80065
  visual:
    visual id:    0x80065
```

```
      class:    PseudoColor
      depth:    8 planes
      size of colormap:    256 entries
      red, green, blue masks:    0x0, 0x0, 0x0
      significant bits in color specification:    8 bits
    visual:
      visual id:    0x80066
      class:    DirectColor
      depth:    8 planes
      size of colormap:    8 entries
      red, green, blue masks:    0x7, 0x38, 0xc0
      significant bits in color specification:    8 bits
    visual:
      visual id:    0x80067
      class:    GrayScale
      depth:    8 planes
      size of colormap:    256 entries
      red, green, blue masks:    0x0, 0x0, 0x0
      significant bits in color specification:    8 bits
    visual:
      visual id:    0x80068
      class:    StaticGray
      depth:    8 planes
      size of colormap:    256 entries
      red, green, blue masks:    0x0, 0x0, 0x0
      significant bits in color specification:    8 bits
    visual:
      visual id:    0x80069
      class:    StaticColor
      depth:    8 planes
      size of colormap:    256 entries
      red, green, blue masks:    0x7, 0x38, 0xc0
      significant bits in color specification:    8 bits
    visual:
      visual id:    0x8006a
      class:    TrueColor
      depth:    8 planes
      size of colormap:    8 entries
      red, green, blue masks:    0x7, 0x38, 0xc0
      significant bits in color specification:    8 bits
    number of mono multibuffer types:    6
      visual id, max buffers, depth:    0x80065, 0, 8
      visual id, max buffers, depth:    0x80066, 0, 8
      visual id, max buffers, depth:    0x80067, 0, 8
      visual id, max buffers, depth:    0x80068, 0, 8
      visual id, max buffers, depth:    0x80069, 0, 8
      visual id, max buffers, depth:    0x8006a, 0, 8
    number of stereo multibuffer types:    0

  screen #1:
    dimensions:    1152x900 pixels (325x254 millimeters)
    resolution:    90x90 dots per inch
    depths (1):    1
```

```
root window id:    0x80070
depth of root window:    1 plane
number of colormaps:    minimum 1, maximum 1
default colormap:    0x8006c
default number of colormap cells:    2
preallocated pixels:    black 1, white 0
options:    backing-store YES, save-unders YES
current input event mask:    0xd0801d
  KeyPressMask            ButtonPressMask         ButtonReleaseMask
  EnterWindowMask         ExposureMask            SubstructureRedirectMask
  PropertyChangeMask      ColormapChangeMask
number of visuals:    1
default visual id:  0x80064
visual:
  visual id:    0x80064
  class:    StaticGray
  depth:    1 plane
  size of colormap:    2 entries
  red, green, blue masks:    0x0, 0x0, 0x0
  significant bits in color specification:    1 bits
number of mono multibuffer types:    1
  visual id, max buffers, depth:    0x80064, 0, 1
number of stereo multibuffer types:    0
```

ENVIRONMENT

DISPLAY

To get the default host, display number, and screen.

SEE ALSO

X(1), xwininfo(1), xprop(1), xrdb(1)

COPYRIGHT

AUTHOR

Jim Fulton, MIT X Consortium

NAME

xedit – simple text editor for X

SYNTAX

xedit [–*toolkitoption* . . .] [*filename*]

DESCRIPTION

Xedit provides a window consisting of the following four areas:

Commands Section	A set of commands that allow you to exit *xedit*, save the file, or load a new file into the edit window.
Message Window	Displays *xedit* messages. In addition, this window can be used as a scratch pad.
Filename Display	Displays the name of the file currently being edited, and whether this file is *Read-Write* or *Read Only*.
Edit Window	Displays the text of the file that you are editing or creating.

OPTIONS

Xedit accepts all of the standard X Toolkit command line options (see *X*(1)). The order of the command line options is not important.

filename Specifies the file that is to be loaded during start-up. This is the file which will be edited. If a file is not specified, *xedit* lets you load a file or create a new file after it has started up.

EDITING

The Athena Text widget is used for the three sections of this application that allow text input. The characters typed will go to the Text widget that the pointer cursor is currently over. If the pointer cursor is not over a text widget then the keypresses will have no effect on the application. This is also true for the special key sequences that popup dialog widgets, so typing Control-S in the filename widget will enable searching in that widget, not the edit widget.

Both the message window and the edit window will create a scrollbar if the text to display is too large to fit in that window. Horizontal scrolling is not allowed by default, but can be turned on through the Text widget's resources, see *Athena Widget Set* for the exact resource definition.

The following keystroke combinations are defined:

Ctrl-a	Beginning Of Line	Meta-b	Backward Word
Ctrl-b	Backward Character	Meta-f	Forward Word
Ctrl-d	Delete Next Character	Meta-i	Insert File
Ctrl-e	End Of Line	Meta-k	Kill To End Of Paragraph
Ctrl-f	Forward Character	Meta-q	Form Paragraph
Ctrl-g	Multiply Reset	Meta-v	Previous Page
Ctrl-h	Delete Previous Character	Meta-y	Insert Current Selection
Ctrl-j	Newline And Indent	Meta-z	Scroll One Line Down
Ctrl-k	Kill To End Of Line	Meta-d	Delete Next Word
Ctrl-l	Redraw Display	Meta-D	Kill Word
Ctrl-m	Newline	Meta-h	Delete Previous Word
Ctrl-n	Next Line	Meta-H	Backward Kill Word
Ctrl-o	Newline And Backup	Meta-<	Beginning Of File
Ctrl-p	Previous Line	Meta->	End Of File
Ctrl-r	Search/Replace Backward	Meta-]	Forward Paragraph
Ctrl-s	Search/Replace Forward	Meta-[	Backward Paragraph
Ctrl-t	Transpose Characters		
Ctrl-u	Multiply by 4	Meta-Delete	Delete Previous Word
Ctrl-v	Next Page	Meta-Shift Delete	Kill Previous Word
Ctrl-w	Kill Selection	Meta-Backspace	Delete Previous Word

Ctrl-y	Unkill	Meta-Shift Backspace	Kill Previous Word
Ctrl-z	Scroll One Line Up		

In addition, the pointer may be used to cut and paste text:

Button 1 Down	Start Selection
Button 1 Motion	Adjust Selection
Button 1 Up	End Selection (cut)
Button 2 Down	Insert Current Selection (paste)
Button 3 Down	Extend Current Selection
Button 3 Motion	Adjust Selection
Button 3 Up	End Selection (cut)

COMMANDS

Quit Quits the current editing session. If any changes have not been saved, *xedit* displays a warning message, allowing the user to save the file.

Save If file backups are enabled (see RESOURCES, below) *xedit* stores a copy of the original, unedited file in <prefix>*file*<suffix>, then overwrites the *file* with the contents of the edit window. The filename is retrieved from the Text widget directly to the right of the *Load* button.

Load Loads the file named in the text widget immediately to the right of the this button and displays it in the Edit window. If the currently displayed file has been modified a warning message will ask the user to save the changes, or press *Load* again.

RESOURCES

For *xedit* the available resources are:

enableBackups (Class **EnableBackups)**

Specifies that, when edits made to an existing file are saved, *xedit* is to copy the original version of that file to <prefix>*file*<suffix> before it saves the changes. The default value for this resource is ‘‘off,’’ stating that no backups should be created.

backupNamePrefix (Class **BackupNamePrefix)**

Specifies a string that is to be prepended to the backup filename. The default is that no string shall be prepended.

backupNameSuffix (Class **BackupNameSuffix)**

Specifies a string that is to be appended to the backup filename. The default is to use ‘‘.BAK’’ as the suffix.

WIDGETS

In order to specify resources, it is useful to know the hierarchy of the widgets which compose *xedit*. In the notation below, indentation indicates hierarchical structure. The widget class name is given first, followed by the widget instance name.

```
Xedit  xedit
        Paned  paned
                Paned  buttons
                        Command  quit
                        Command  save
                        Command  load
                        Text  filename
                Label  bc_label
                Text  messageWindow
                Label  labelWindow
```

Text editWindow

ENVIRONMENT

DISPLAY to get the default host and display number.

XENVIRONMENT to get the name of a resource file that overrides the global resources stored in the RESOURCE_MANAGER property.

FILES

/usr/lib/X11/app-defaults/Xedit
specifies required resources

SEE ALSO

X(1), *xrdb*(1), *Athena Widget Set*

RESTRICTIONS

There is no *undo* function.

COPYRIGHT

AUTHOR

Chris D. Peterson, MIT X Consortium

NAME

xev - print contents of X events

SYNOPSIS

xev [–display *displayname*] [–geometry *geom*] [–bw *pixels*] [–bs *{NotUseful,WhenMapped,Always}*] [–id *windowid*] [–s] [–name *string*] [–rv]

DESCRIPTION

Xev creates a window and then asks the X server to send it *events* whenever anything happens to the window (such as being moved, resized, typed in, clicked in, etc.). You can also attach it to an existing window. It is useful for seeing what causes events to occur and to display the information that they contain.

OPTIONS

–display *display*
This option specifies the X server to contact.

–geometry *geom*
This option specifies the size and/or location of the window.

–bw *pixels*
This option specifies the border width for the window.

–bs *{NotUseful,WhenMapped,Always}*
This option specifies what kind of backing store to give the window. The default is NotUseful.

–id *windowid*
This option specifies that the window with the given id should be monitored, instead of creating a new window.

–s This option specifies that save-unders should be enabled on the window.

–name *string*
This option specifies the name to assign to the created window.

–rv This option specifies that the window should be in reverse video.

SEE ALSO

X(1), xwininfo(1), xdpyinfo(1), Xlib Programmers Manual, X Protocol Specification

COPYRIGHT

AUTHOR

Jim Fulton, MIT X Consortium

NAME

xeyes – a follow the mouse X demo

SYNOPSIS

xeyes [-option ...]

DESCRIPTION

Xeyes watches what you do and reports to the Boss.

OPTIONS

–fg *foreground color*
: choose a different color for the pupil of the eyes.

–bg *background color*
: choose a different color for the background.

–outline *outline color*
: choose a different color for the outline of the eyes.

–center *center color*
: choose a different color for the center of the eyes.

–backing *{ WhenMapped Always NotUseful }*
: selects an appropriate level of backing store.

–geometry *geometry*
: define the initial window geometry; see *X(1)*.

–display *display*
: specify the display to use; see *X(1)*.

–bd *border color*
: choose a different color for the window border.

–bw *border width*
: choose a different width for the window border.

–shape uses the SHAPE extension to shape the window. This is the default.

+shape Disables uses the SHAPE extension to shape the window.

SEE ALSO

X(1), X Toolkit documentation

COPYRIGHT

AUTHOR

Keith Packard, MIT X Consortium
Copied from the NeWS version written (apparantly) by Jeremy Huxtable as seen at SIGGRAPH ’88

NAME

xfd – display all the characters in an X font

SYNOPSIS

xfd [–options ...] –fn *fontname*

DESCRIPTION

The *xfd* utility creates a window containing the name of the font being displayed, a row of command buttons, several lines of text for displaying character metrics, and a grid containing one glyph per cell. The characters are shown in increasing order from left to right, top to bottom. The first character displayed at the top left will be character number 0 unless the **–start** option has been supplied in which case the character with the number given in the **–start** option will be used.

The characters are displayed in a grid of boxes, each large enough to hold any single character in the font. Each character glyph is drawn using the PolyText16 request (used by the *Xlib* routine **XDrawString16**). If the **–box** option is given, a rectangle will be drawn around each character, showing where an ImageText16 request (used by the *Xlib* routine **XDrawImageString16**) would cause background color to be displayed.

The origin of each glyph is normally set so that the character is drawn in the upper left hand corner of the grid cell. However, if a glyph has a negative left bearing or an unusually large ascent, descent, or right bearing (as is the case with *cursor* font), some character may not appear in their own grid cells. The **–center** option may be used to force all glyphs to be centered in their respective cells.

All the characters in the font may not fit in the window at once. To see the next page of glyphs, press the *Next* button at the top of the window. To see the previous page, press *Prev*. To exit *xfd*, press *Quit*.

Individual character metrics (index, width, bearings, ascent and descent) can be displayed at the top of the window by pressing on the desired character.

The font name displayed at the top of the window is the full name of the font, as determined by the server. See *xlsfonts* for ways to generate lists of fonts, as well as more detailed summaries of their metrics and properties.

OPTIONS

xfd accepts all of the standard toolkit command line options along with the additional options listed below:

–fn *font* This option specifies the font to be displayed.

–box This option indicates that a box should be displayed outlining the area that would be filled with background color by an ImageText request.

–center This option indicates that each glyph should be centered in its grid.

–start *number*
This option specifies the glyph index of the upper left hand corner of the grid. This is used to view characters at arbitrary locations in the font. The default is 0.

–bc *color*
This option specifies the color to be used if ImageText boxes are drawn.

X DEFAULTS

To be written.

SEE ALSO

X(1), xlsfonts(1), xrdb(1), xfontsel(1)

BUGS

The program should skip over pages full of non-existent characters.

COPYRIGHT

AUTHOR

Jim Fulton, MIT X Consortium; previous program of the same name by Mark Lillibridge, MIT Project Athena.

NAME

xfontsel - point & click interface for selecting X11 font names

SYNTAX

xfontsel [-*toolkitoption* ...] [**-pattern** *fontname*] **[-print] [-sample** *text*] **[-sample16** *text16*] **[-noscaled]**

DESCRIPTION

The *xfontsel* application provides a simple way to display the fonts known to your X server, examine samples of each, and retrieve the X Logical Font Description ("XLFD") full name for a font.

If **-pattern** is not specified, all fonts with XLFD 14-part names will be selectable. To work with only a subset of the fonts, specify **-pattern** followed by a partially or fully qualified font name; e.g., ‘‘-pattern *medium*’’ will select that subset of fonts which contain the string ‘‘medium’’ somewhere in their font name. Be careful about escaping wildcard characters in your shell.

If **-print** is specified on the command line the selected font specifier will be written to standard output when the *quit* button is activated. Regardless of whether or not **-print** was specified, the font specifier may be made the PRIMARY (text) selection by activating the *select* button.

The **-sample** option specifies the sample text to be used to display the selected font if the font is linearly indexed, overriding the default.

The **-sample16** option specifies the sample text to be used to display the selected font if the font is matrix encoded, overriding the default.

The **-noscaled** option disables the ability to select scaled fonts at arbitrary pixel or point sizes. This makes it clear which bitmap sizes are advertised by the server, and can avoid an accidental and sometimes prolonged wait for a font to be scaled.

INTERACTIONS

Clicking any pointer button in one of the XLFD field names will pop up a menu of the currently-known possibilities for that field. If previous choices of other fields were made, only values for fonts which matched the previously selected fields will be selectable; to make other values selectable, you must deselect some other field(s) by choosing the ‘‘*’’ entry in that field. Unselectable values may be omitted from the menu entirely as a configuration option; see the **ShowUnselectable** resource, below. Whenever any change is made to a field value, *xfontsel* will assert ownership of the PRIMARY_FONT selection. Other applications (see, e.g., *xterm*) may then retrieve the selected font specification.

Scalable fonts come back from the server with zero for the pixel size, point size, and average width fields. Selecting a font name with a zero in these positions results in an implementation-dependent size. Any pixel or point size can be selected to scale the font to a particular size. Any average width can be selected to anamorphically scale the font (although you may find this challenging given the size of the average width menu).

Clicking the left pointer button in the *select* widget will cause the currently selected font name to become the PRIMARY text selection as well as the PRIMARY_FONT selection. This then allows you to paste the string into other applications. The **select** button remains highlighted to remind you of this fact, and de-highlights when some other application takes the PRIMARY selection away. The *select* widget is a toggle; pressing it when it is highlighted will cause *xfontsel* to release the selection ownership and de-highlight the widget. Activating the *select* widget twice is the only way to cause *xfontsel* to release the PRIMARY_FONT selection.

RESOURCES

The application class is **XFontSel**. Most of the user-interface is configured in the app-defaults file; if this file is missing a warning message will be printed to standard output and the resulting window will be nearly incomprehensible.

Most of the significant parts of the widget hierarchy are documented in the app-defaults file (normally /usr/lib/X11/app-defaults/XFontSel).

Application specific resources:

cursor (class **Cursor)**
Specifies the cursor for the application window.

pattern (class **Pattern)**
Specifies the font name pattern for selecting a subset of available fonts. Equivalent to the **-pattern** option. Most useful patterns will contain at least one field delimiter; e.g. ‘‘*-m-*’’ for monospaced fonts.

pixelSizeList (class **PixelSizeList)**
Specifies a list of pixel sizes to add to the pixel size menu, so that scalable fonts can be selected at those pixel sizes. The default pixelSizeList contains 7, 30, 40, 50, and 60.

pointSizeList (class **PointSizeList)**
Specifies a list of point sizes (in units of tenths of points) to add to the point size menu, so that scalable fonts can be selected at those point sizes. The default pointSizeList contains 250, 300, 350, and 400.

printOnQuit (class **PrintOnQuit)**
If *True* the currently selected font name is printed to standard output when the quit button is activated. Equivalent to the **-print** option.

sampleText (class **Text)**
The sample 1-byte text to use for linearly indexed fonts. Each glyph index is a single byte, with newline separating lines.

sampleText16 (class **Text16)**
The sample 2-byte text to use for matrix-encoded fonts. Each glyph index is two bytes, with a 1-byte newline separating lines.

scaledFonts (class **ScaledFonts)**
If *True* then selection of arbitrary pixel and point sizes for scalable fonts is enabled.

Widget specific resources:

showUnselectable (class **ShowUnselectable)**
Specifies, for each field menu, whether or not to show values that are not currently selectable, based upon previous field selections. If shown, the unselectable values are clearly identified as such and do not highlight when the pointer is moved down the menu. The full name of this resource is **fieldN.menu.options.showUnselectable**, class **MenuButton.SimpleMenu.Options.ShowUnselectable**; where N is replaced with the field number (starting with the left-most field numbered 0). The default is True for all but field 11 (average width of characters in font) and False for field 11. If you never want to see unselectable entries, ’*menu.options.showUnselectable:False’ is a reasonable thing to specify in a resource file.

FILES

$XFILESEARCHPATH/XFontSel

SEE ALSO

xrdb(1), xfd(1)

BUGS

Sufficiently ambiguous patterns can be misinterpreted and lead to an initial selection string which may not correspond to what the user intended and which may cause the initial sample text output to fail to match the proffered string. Selecting any new field value will correct the sample output, though possibly resulting in no matching font.

Should be able to return a FONT for the PRIMARY selection, not just a STRING.

Any change in a field value will cause *xfontsel* to assert ownership of the PRIMARY_FONT selection. Perhaps this should be parameterized.

When running on a slow machine, it is possible for the user to request a field menu before the font names have been completely parsed. An error message indicating a missing menu is printed to stderr but otherwise nothing bad (or good) happens.

The average-width menu is too large to be useful.

COPYRIGHT

AUTHOR

Ralph R. Swick, Digital Equipment Corporation/MIT Project Athena

NAME

xgas - animated simulation of an ideal gas

SYNOPSIS

xgas [-option ...]

DESCRIPTION

xgas is a physical simulation of an ideal gas in a heated box. Gas molecules move around the box with velocities dependent on their temperature. A chamber consisting of two boxes contains the gas molecules; the temperature of each box can be independently controlled by a scrollbar. When gas molecules collide with the walls, their temperature approaches that of the box.

Use mouse button 1 to create molecules one at a time at the cursor position. Use mouse button 2 to create the maximum number of molecules at the cursor position.

OPTIONS

This is a toolkit program;
all the usual toolkit options are available.

X DEFAULTS

The *xgas* program uses the following X resources:

timeStepSize
Specifies the simulated time duration in microseconds for each cycle of computation.

delay Specifies the real time interval between timestep computations.

randomBounce
In each wall collision, a molecule bounces elastically (angle of incidence = angle of reflection). A component of randomness is added to this angle. RandomBounce varies from 0.0 (no randomness) to 1.0 (completely random angle of incidence).

equilibrium
During each wall collision, a molecule's kinetic energy approaches that corresponding to the temperature of the wall. If equilibrium is 1.0, the molecule reaches the wall temperature immediately. For values between 1.0 and 0.0, the molecule approaches the temperature of the wall more slowly.

maxMolecules
Specifies the maximum number of molecules in the simulation.

SEE ALSO

X(1), xwd(1)

BUGS

When the chamber is resized, molecules should be rearranged appropriately. Instead, the molecule arrays are reinitialized.

COPYRIGHT

AUTHOR

Larry Medwin

NAME

xgc - X graphics demo

SYNOPSIS

xgc [-*toolkitoption* ...]

DESCRIPTION

The *xgc* program demonstrates various features of the X graphics primitives. Try the buttons, see what they do; we haven't the time to document them, perhaps you do?

OPTIONS

Xgc accepts all of the standard X Toolkit command line options.

X DEFAULTS

This program accepts the usual defaults for toolkit applications.

ENVIRONMENT

DISPLAY

to get the default host and display number.

XENVIRONMENT

to get the name of a resource file that overrides the global resources stored in the RESOURCE_MANAGER property.

SEE ALSO

X(1)

BUGS

This program isn't really finished yet.

COPYRIGHT

AUTHORS

Dan Schmidt, MIT

NAME

xhost - server access control program for X

SYNOPSIS

xhost [[+-]name ...]

DESCRIPTION

The *xhost* program is used to add and delete host names or user names to the list allowed to make connections to the X server. In the case of hosts, this provides a rudimentary form of privacy control and security. It is only sufficient for a workstation (single user) environment, although it does limit the worst abuses. Environments which require more sophisticated measures should implement the user-based mechanism, or use the hooks in the protocol for passing other authentication data to the server.

Hostnames that are followed by two colons (::) are used in checking DECnet connections; all other hostnames are used for TCP/IP connections.

User names contain an at-sign (@). When Secure RPC is being used, the network independent netname (e.g., "unix.*uid@domainname*") can be specified, or a local user can be specified with just the username and a trailing at-sign (e.g., "joe@").

OPTIONS

Xhost accepts the following command line options described below. For security, the options that effect access control may only be run from the "controlling host". For workstations, this is the same machine as the server. For X terminals, it is the login host.

[+]*name* The given *name* (the plus sign is optional) is added to the list allowed to connect to the X server. The name can be a host name or a user name.

–*name* The given *name* is removed from the list of allowed to connect to the server. The name can be a host name or a user name. Existing connections are not broken, but new connection attempts will be denied. Note that the current machine is allowed to be removed; however, further connections (including attempts to add it back) will not be permitted. Resetting the server (thereby breaking all connections) is the only way to allow local connections again.

+ Access is granted to everyone, even if they aren’t on the list (i.e., access control is turned off).

– Access is restricted to only those on the list (i.e., access control is turned on).

nothing If no command line arguments are given, a message indicating whether or not access control is currently enabled is printed, followed by the list of those allowed to connect. This is the only option that may be used from machines other than the controlling host.

DIAGNOSTICS

For each name added to the access control list, a line of the form "*name* being added to access contro list" is printed. For each name removed from the access control list, a line of the form "*name* being removed from access contro list" is printed.

FILES

/etc/X*.hosts

SEE ALSO

X(1), Xserver(1), xdm(1)

ENVIRONMENT

DISPLAY

to get the default host and display to use.

BUGS

You can’t specify a display on the command line because **–display** is a valid command line argument (indicating that you want to remove the machine named *‘‘display’’* from the access list).

This is not really a bug, but the X server stores network addresses, not host names. If somehow you change a host's network address while the server is still running, *xhost* must be used to add the new address and/or remove the old address.

COPYRIGHT

AUTHORS

Bob Scheifler, MIT Laboratory for Computer Science,
Jim Gettys, MIT Project Athena (DEC).

NAME

xinit – X Window System initializer

SYNOPSIS

xinit [[*client*] *options*] [**––** [*server*] [*display*] *options*]

DESCRIPTION

The *xinit* program is used to start the X Window System server and a first client program on systems that cannot start X directly from */etc/init* or in environments that use multiple window systems. When this first client exits, *xinit* will kill the X server and then terminate.

If no specific client program is given on the command line, *xinit* will look for a file in the user's home directory called *.xinitrc* to run as a shell script to start up client programs. If no such file exists, *xinit* will use the following as a default:

```
xterm –geometry +1+1 –n login –display :0
```

If no specific server program is given on the command line, *xinit* will look for a file in the user's home directory called *.xserverrc* to run as a shell script to start up the server. If no such file exists, *xinit* will use the following as a default:

```
X :0
```

Note that this assumes that there is a program named *X* in the current search path. However, servers are usually named *Xdisplaytype* where *displaytype* is the type of graphics display which is driven by this server. The site administrator should, therefore, make a link to the appropriate type of server on the machine, or create a shell script that runs *xinit* with the appropriate server.

An important point is that programs which are run by *.xinitrc* should be run in the background if they do not exit right away, so that they don't prevent other programs from starting up. However, the last long-lived program started (usually a window manager or terminal emulator) should be left in the foreground so that the script won't exit (which indicates that the user is done and that *xinit* should exit).

An alternate client and/or server may be specified on the command line. The desired client program and its arguments should be given as the first command line arguments to *xinit*. To specify a particular server command line, append a double dash (––) to the *xinit* command line (after any client and arguments) followed by the desired server command.

Both the client program name and the server program name must begin with a slash (/) or a period (.). Otherwise, they are treated as an arguments to be appended to their respective startup lines. This makes it possible to add arguments (for example, foreground and background colors) without having to retype the whole command line.

If an explicit server name is not given and the first argument following the double dash (––) is a colon followed by a digit, *xinit* will use that number as the display number instead of zero. All remaining arguments are appended to the server command line.

EXAMPLES

Below are several examples of how command line arguments in *xinit* are used.

xinit This will start up a server named *X* and run the user's *.xinitrc*, if it exists, or else start an *xterm*.

xinit –– /usr/bin/X11/Xqdss :1
This is how one could start a specific type of server on an alternate display.

xinit –geometry =80x65+10+10 –fn 8x13 –j –fg white –bg navy
This will start up a server named *X*, and will append the given arguments to the default *xterm* command. It will ignore *.xinitrc*.

xinit –e widgets –– ./Xsun –l –c

This will use the command *./Xsun –l –c* to start the server and will append the arguments *–e widgets* to the default *xterm* command.

xinit /usr/ucb/rsh fasthost cpupig –display ws:1 –– :1 –a 2 –t 5
This will start a server named *X* on display 1 with the arguments *–a 2 –t 5*. It will then start a remote shell on the machine **fasthost** in which it will run the command *cpupig*, telling it to display back on the local workstation.

Below is a sample *.xinitrc* that starts a clock, several terminals, and leaves the window manager running as the ‘‘last’’ application. Assuming that the window manager has been configured properly, the user then chooses the ‘‘Exit’’ menu item to shut down X.

```
xrdb –load $HOME/.Xresources
xsetroot –solid gray &
xclock –g 50x50–0+0 –bw 0 &
xload –g 50x50–50+0 –bw 0 &
xterm –g 80x24+0+0 &
xterm –g 80x24+0–0 &
twm
```

Sites that want to create a common startup environment could simply create a default *.xinitrc* that references a site-wide startup file:

```
#!/bin/sh
. /usr/local/lib/site.xinitrc
```

Another approach is to write a script that starts *xinit* with a specific shell script. Such scripts are usually named *x11*, *xstart*, or *startx* and are a convenient way to provide a simple interface for novice users:

```
#!/bin/sh
xinit /usr/local/lib/site.xinitrc –– /usr/bin/X11/X bc
```

ENVIRONMENT VARIABLES

DISPLAY This variable gets set to the name of the display to which clients should connect.

XINITRC This variable specifies an init file containing shell commands to start up the initial windows. By default, *.xinitrc* in the home directory will be used.

FILES

.xinitrc default client script

xterm client to run if *.xinitrc* does not exist

.xserverrc default server script

X server to run if *.xserverrc* does not exist

SEE ALSO

X(1), *startx*(1), *Xserver*(1), *xterm*(1)

COPYRIGHT

AUTHOR

Bob Scheifler, MIT Laboratory for Computer Science

NAME

xkill - kill a client by its X resource

SYNOPSIS

xkill [–display *displayname*] [–id *resource*] [–button number] [–frame] [–all]

DESCRIPTION

Xkill is a utility for forcing the X server to close connections to clients. This program is very dangerous, but is useful for aborting programs that have displayed undesired windows on a user's screen. If no resource identifier is given with *-id*, *xkill* will display a special cursor as a prompt for the user to select a window to be killed. If a pointer button is pressed over a non-root window, the server will close its connection to the client that created the window.

OPTIONS

–display *displayname*
: This option specifies the name of the X server to contact.

–id *resource*
: This option specifies the X identifier for the resource whose creator is to be aborted. If no resource is specified, *xkill* will display a special cursor with which you should select a window to be kill.

–button *number*
: This option specifies the number of pointer button that should be used in selecting a window to kill. If the word "any" is specified, any button on the pointer may be used. By default, the first button in the pointer map (which is usually the leftmost button) is used.

–all This option indicates that all clients with top-level windows on the screen should be killed. *Xkill* will ask you to select the root window with each of the currently defined buttons to give you several chances to abort. Use of this option is highly discouraged.

–frame This option indicates that xkill should ignore the standard conventions for finding top-level client windows (which are typically nested inside a window manager window), and simply believe that you want to kill direct children of the root.

XDEFAULTS

Button Specifies a specific pointer button number or the word "any" to use when selecting windows.

SEE ALSO

X(1), xwininfo(1), XKillClient and XGetPointerMapping in the Xlib Programmers Manual, KillClient in the X Protocol Specification

COPYRIGHT

AUTHOR

Jim Fulton, MIT X Consortium
Dana Chee, Bellcore

NAME

xload – system load average display for X

SYNOPSIS

xload [-*toolkitoption* ...] [-scale *integer*] [-update *seconds*] [-hl *color*] [-highlight *color*] [-jumpscroll *pixels*] [-label *string*] [-nolabel] [-lights]

DESCRIPTION

The *xload* program displays a periodically updating histogram of the system load average.

OPTIONS

Xload accepts all of the standard X Toolkit command line options (see *X(1)*). The order of the options in unimportant. *xload also accepts the following additional options:*

–hl *color* **or –highlight** *color*
: This option specifies the color of the scale lines.

–jumpscroll number of pixels
: The number of pixels to shift the graph to the left when the graph reaches the right edge of the window. The default value is 1/2 the width of the current window. Smooth scrolling can be achieved by setting it to 1.

–label *string*
: The string to put into the label above the load average.

–nolabel If this command line option is specified then no label will be displayed above the load graph.

–lights When specified, this option causes *xload* to display the current load average by using the keyboard leds; for a load average of *n*, xload lights the first *n* keyboard leds. This option turns off the usual screen display.

–scale *integer*
: This option specifies the minimum number of tick marks in the histogram, where one division represents one load average point. If the load goes above this number, *xload* will create more divisions, but it will never use fewer than this number. The default is 1.

–update *seconds*
: This option specifies the interval in seconds at which *xload* updates its display. The minimum amount of time allowed between updates is 1 second. The default is 10.

RESOURCES

In addition to the resources available to each of the widgets used by *xload* there is one resource defined by the application itself.

showLabel (class **Boolean)**
: If False then no label will be displayed.

WIDGETS

In order to specify resources, it is useful to know the hierarchy of the widgets which compose *xload*. In the notation below, indentation indicates hierarchical structure. The widget class name is given first, followed by the widget instance name.

```
XLoad  xload
        Paned  paned
                Label  label
                StripChart  load
```

ENVIRONMENT

DISPLAY
: to get the default host and display number.

XENVIRONMENT
to get the name of a resource file that overrides the global resources stored in the RESOURCE_MANAGER property.

FILES

/usr/lib/X11/app-defaults/XLoad - specifies required resources

SEE ALSO

X(1), xrdb(1), mem(4), Athena StripChart Widget.

BUGS

This program requires the ability to open and read the special system file */dev/kmem*. Sites that do not allow general access to this file should make *xload* belong to the same group as */dev/kmem* and turn on the *set group id* permission flag.

Reading /dev/kmem is inherently non-portable. Therefore, the routine used to read it (get_load.c) must be ported to each new operating system.

COPYRIGHT

AUTHORS

K. Shane Hartman (MIT-LCS) and Stuart A. Malone (MIT-LCS);
with features added by Jim Gettys (MIT-Athena), Bob Scheifler (MIT-LCS), Tony Della Fera (MIT-Athena), and Chris Peterson (MIT-LCS).

NAME

xlogo - X Window System logo

SYNOPSIS

xlogo [-*toolkitoption* ...]

DESCRIPTION

The *xlogo* program displays the X Window System logo. This program is nothing more than a wrapper around the *undocumented* Athena Logo widget.

OPTIONS

Xlogo accepts all of the standard X Toolkit command line options, as well as the following:

–shape This option indicates that the logo window should be shaped rather than rectangular.

RESOURCES

The default width and the default height are each 100 pixels. This program uses the *Logo* widget in the Athena widget set. It understands all of the Simple widget resource names and classes as well as:

foreground (class **Foreground)**
: Specifies the color for the logo. The default is depends on whether *reverseVideo* is specified. If *reverseVideo* is specified the default is *XtDefaultForeground*, otherwise the default is *XtDefault-Background*.

shapeWindow (class **ShapeWindow)**
: Specifies that the window is shaped to the X logo. The default is False.

WIDGETS

In order to specify resources, it is useful to know the hierarchy of the widgets which compose *xlogo*. In the notation below, indentation indicates hierarchical structure. The widget class name is given first, followed by the widget instance name.

```
XLogo  xlogo
        Logo  xlogo
```

ENVIRONMENT

DISPLAY
: to get the default host and display number.

XENVIRONMENT
: to get the name of a resource file that overrides the global resources stored in the RESOURCE_MANAGER property.

FILES

/usr/lib/X11/app-defaults/XLogo - specifies required resources

SEE ALSO

X(1), xrdb(1)

COPYRIGHT

AUTHORS

Ollie Jones of Apollo Computer and Jim Fulton of the MIT X Consortium wrote the logo graphics routine, based on a graphic design by Danny Chong and Ross Chapman of Apollo Computer.

NAME

xlsatoms - list interned atoms defined on server

SYNOPSIS

xlsatoms [-options ...]

DESCRIPTION

Xlsatoms lists the interned atoms. By default, all atoms starting from 1 (the lowest atom value defined by the protocol) are listed until unknown atom is found. If an explicit range is given, *xlsatoms* will try all atoms in the range, regardless of whether or not any are undefined.

OPTIONS

–display *dpy*
This option specifies the X server to which to connect.

–format *string*
This option specifies a *printf*-style string used to list each atom <*value*,*name*> pair, printed in that order (*value* is an *unsigned long* and *name* is a *char* ∗). *Xlsatoms* will supply a newline at the end of each line. The default is *%ld\t%s*.

–range *[low]-[high]*
This option specifies the range of atom values to check. If *low* is not given, a value of 1 assumed. If *high* is not given, *xlsatoms* will stop at the first undefined atom at or above *low*.

–name *string*
This option specifies the name of an atom to list. If the atom does not exist, a message will be printed on the standard error.

SEE ALSO

X(1), Xserver(1), xprop(1)

ENVIRONMENT

DISPLAY
to get the default host and display to use.

COPYRIGHT

AUTHOR

Jim Fulton, MIT X Consortium

NAME

xlsclients - list client applications running on a display

SYNOPSIS

xlsclients [-display *displayname*] [-a] [-l] [-m maxcmdlen]

DESCRIPTION

Xlsclients is a utility for listing information about the client applications running on a display. It may be used to generate scripts representing a snapshot of the the user's current session.

OPTIONS

–display *displayname*
: This option specifies the X server to contact.

–a
: This option indicates that clients on all screens should be listed. By default, only those clients on the default screen are listed.

–l
: This option indicates that a long listing showing the window name, icon name, and class hints in addition to the machine name and command string shown in the default listing.

–m *maxcmdlen*
: This option specifies the maximum number of characters in a command to print out. The default is 10000.

ENVIRONMENT

DISPLAY
: To get the default host, display number, and screen.

SEE ALSO

X(1), xwininfo(1), xprop(1)

COPYRIGHT

AUTHOR

Jim Fulton, MIT X Consortium

NAME

xlsfonts - server font list displayer for X

SYNOPSIS

xlsfonts [-options ...] [-fn pattern]

DESCRIPTION

Xlsfonts lists the fonts that match the given *pattern*. The wildcard character "∗" may be used to match any sequence of characters (including none), and "?" to match any single character. If no pattern is given, "∗" is assumed.

The "∗" and "?" characters must be quoted to prevent them from being expanded by the shell.

OPTIONS

–display *host***:***dpy*
This option specifies the X server to contact.

–l[l[l]] This option indicates that medium, long, and very long listings, respectively, should be generated for each font.

–m This option indicates that long listings should also print the minimum and maximum bounds of each font.

–C This option indicates that listings should use multiple columns. This is the same as **-n 0**.

–1 This option indicates that listings should use a single column. This is the same as **-n 1**.

–w *width*
This option specifies the width in characters that should be used in figuring out how many columns to print. The default is 79.

–n *columns*
This option specifies the number of columns to use in displaying the output. By default, it will attempt to fit as many columns of font names into the number of character specified by **-w** *width*.

–u This option indicates that the output should be left unsorted.

–o This option indicates that *xlsfonts* should do an **OpenFont** (and **QueryFont**, if appropriate) rather than a **ListFonts**. This is useful if **ListFonts** or **ListFontsWithInfo** fail to list a known font (as is the case with some scaled font systems).

SEE ALSO

X(1), Xserver(1), xset(1), xfd(1)

ENVIRONMENT

DISPLAY
to get the default host and display to use.

BUGS

Doing ‘‘xlsfonts -l’’ can tie up your server for a very long time. This is really a bug with single-threaded non-preemptable servers, not with this program.

COPYRIGHT

AUTHOR

Mark Lillibridge, MIT Project Athena; Jim Fulton, MIT X Consortium; Phil Karlton, SGI

NAME

xmag **–** magnify parts of the screen

SYNOPSIS

xmag [**–mag** *magfactor*] [**–source** *geom*] [*–toolkitoption* . . .]

DESCRIPTION

The *xmag* program allows you to magnify portions of an X screen. If no explicit region is specified, a square with the pointer in the upper left corner is displayed indicating the area to be enlarged. The area can be dragged out to the desired size by pressing Button 2. Once a region has been selected, a window is popped up showing a blown up version of the region in which each pixel in the source image is represented by a small square of the same color. Pressing Button1 in the enlargement window shows the position and RGB value of the pixel under the pointer until the button is released. Typing ‘‘Q’’ or ‘‘^C’’ in the enlargement window exits the program. The application has 5 buttons across its top. *Close* deletes this particular magnification instance. *Replace* brings up the rubber band selector again to select another region for this magnification instance. *New* brings up the rubber band selector to create a new magnification instance. *Cut* puts the magnification image into the primary selection. *Paste* copies the primary selection buffer into *xmag*. Note that you can cut and paste between *xmag* and the *bitmap* program. Resizing *xmag* resizes the magnification area. *xmag* preserves the colormap, visual, and window depth of the source.

WIDGETS

xmag uses the X Toolkit and the Athena Widget Set. The magnified image is displayed in the Scale widget. For more information, see the Athena Widget Set documentation. Below is the widget structure of the *xmag* application. Indentation indicates hierarchical structure. The widget class name is given first, followed by the widget instance name.

```
Xmag xmag
    RootWindow root
    TopLevelShell xmag
        Paned pane1
            Paned pane2
                Command close
                Command replace
                Command new
                Command select
                Command paste
                Label xmag label
            Paned pane2
                Scale scale
    OverrideShell pixShell
        Label pixLabel
```

OPTIONS

–source *geom* This option specifies the size and/or location of the source region on the screen. By default, a 64x64 square is provided for the user to select an area of the screen.

–mag *integer* This option indicates the magnification to be used. 5 is the default.

COPYRIGHT

AUTHORS

Dave Sternlicht and Davor Matic, MIT X Consortium.

NAME

xman – Manual page display program for the X Window System

SYNOPSIS

xman [–*options* . . .]

DESCRIPTION

Xman is a manual page browser. The default size of the initial *xman* window is small so that you can leave it running throughout your entire login session. In the initial window there are three options: *Help* will pop up a window with on-line help, *Quit* will exit, and *Manual Page* will pop up a window with a manual page browser in it. Typing Control-S will pop up a window prompting for a specific manual page to display. You may display more than one manual page browser window at a time from a single execution of *xman*.

For further information on using *xman*, please read the on-line help information. Most of this manual will discuss customization of *xman*.

OPTIONS

Xman supports all standard Toolkit command line arguments (see *X*(1)). The following additional arguments are supported.

–helpfile *filename*
Specifies a helpfile to use other than the default.

–bothshown
Allows both the manual page and manual directory to be on the screen at the same time.

–notopbox
Starts without the Top Menu with the three buttons in it.

–geometry *WxH+X+Y*
Sets the size and location of the Top Menu with the three buttons in it.

–pagesize *WxH+X+Y*
Sets the size and location of all the Manual Pages.

CUSTOMIZING XMAN

Xman allows customization of both the directories to be searched for manual pages, and the name that each directory will map to in the *Sections* menu. Xman determines which directories it will search by reading the *MANPATH* environment variable. If no *MANPATH* is found then the directory is /usr/man is searched on POSIX systems. This environment is expected to be a colon-separated list of directories for xman to search.

setenv MANPATH /mit/kit/man:/usr/man

By default, *xman* will search each of the following directories (in each of the directories specified in the users MANPATH) for manual pages. If manual pages exist in that directory then they are added to list of manual pages for the corresponding menu item. A menu item is only displayed for those sections that actually contain manual pages.

Directory	Section Name
---------	------------
man1	(1) User Commands
man2	(2) System Calls
man3	(3) Subroutines
man4	(4) Devices
man5	(5) File Formats
man6	(6) Games
man7	(7) Miscellaneous
man8	(8) Sys. Administration

manl	(l) Local
mann	(n) New
mano	(o) Old

For instance, a user has three directories in her manual path and each contain a directory called *man3*. All these manual pages will appear alphabetically sorted when the user selects the menu item called *(3) Subroutines*. If there is no directory called *mano* in any of the directories in her MANPATH, or there are no manual pages in any of the directories called *mano* then no menu item will be displayed for the section called *(o) Old*.

THE MANDESC FILE

By using the *mandesc* file a user or system manager is able to more closely control which manual pages will appear in each of the sections represented by menu items in the *Sections* menu. This functionality is only available on a section by section basis, and individual manual pages may not be handled in this manner. (Although generous use of symbolic links — see *ln*(1) — will allow almost any configuration you can imagine.)

The format of the mandesc file is a character followed by a label. The character determines which of the sections will be added under this label. For instance suppose that you would like to create an extra menu item that contains all programmer subroutines. This label should contain all manual pages in both sections two and three. The *mandesc* file would look like this:

```
2Programmer Subroutines
3Programmer Subroutines
```

This will add a menu item to the *Sections* menu that would bring up a listing of all manual pages in sections two and three of the Programmers Manual. Since the label names are *exactly* the same they will be added to the same section. Note, however, that the original sections still exist.

If you want to completely ignore the default sections in a manual directory then add the line:

```
no default sections
```

anywhere in your mandesc file. This keeps xman from searching the default manual sections *In that directory only*. As an example, suppose you want to do the same thing as above, but you don't think that it is useful to have the *System Calls* or *Subroutines* sections any longer. You would need to duplicate the default entries, as well as adding your new one.

```
no default sections
1(1) User Commands
2Programmer Subroutines
3Programmer Subroutines
4(4) Devices
5(5) File Formats
6(6) Games
7(7) Miscellaneous
8(8) Sys. Administration
l(l) Local
n(n) New
o(o) Old
```

Xman will read any section that is of the from *man<character>*, where <character> is an upper or lower case letter (they are treated distinctly) or a numeral (0-9). Be warned, however, that man(1) and catman(8) will not search directories that are non-standard.

WIDGETS

In order to specify resources, it is useful to know the hierarchy of the widgets which compose *xman*. In the notation below, indentation indicates hierarchical structure. The widget class name is given first, followed by the widget instance name.

```
Xman xman       (This widget is never used)
    TopLevelShell  topbox
        Form  form
            Label  topLabel
            Command  helpButton
            Command  quitButton
            Command  manpageButton
        TransientShell  search
            DialogWidgetClass  dialog
                Label  label
                Text  value
                Command  manualPage
                Command  apropos
                Command  cancel
        TransientShell  pleaseStandBy
            Label  label
    TopLevelShell  manualBrowser
        Paned  Manpage_Vpane
            Paned  horizPane
                MenuButton  options
                MenuButton  sections
                Label  manualBrowser
            Viewport  directory
                List  directory
                List  directory
                .
                . (one for each section,
                .  created on the fly)
                .
            ScrollByLine  manualPage
        SimpleMenu  optionMenu
            SmeBSB  displayDirectory
            SmeBSB  displayManualPage
            SmeBSB  help
            SmeBSB  search
            SmeBSB  showBothScreens
            SmeBSB  removeThisManpage
            SmeBSB  openNewManpage
            SmeBSB  showVersion
            SmeBSB  quit
        SimpleMenu  sectionMenu
            SmeBSB  <name of section>
                .
                . (one for each section)
                .
        TransientShell  search
            DialogWidgetClass  dialog
                Label  label
```

```
                        Text  value
                        Command  manualPage
                        Command  apropos
                        Command  cancel
        TransientShell  pleaseStandBy
                Label  label
        TransientShell  likeToSave
                Dialog  dialog
                        Label  label
                        Text  value
                        Command  yes
                        Command  no
TopLevelShell  help
        Paned  Manpage_Vpane
                Paned  horizPane
                        MenuButton  options
                        MenuButton  sections
                        Label  manualBrowser
                ScrollByLine  manualPage
        SimpleMenu  optionMenu
                SmeBSB  displayDirectory
                SmeBSB  displayManualPage
                SmeBSB  help
                SmeBSB  search
                SmeBSB  showBothScreens
                SmeBSB  removeThisManpage
                SmeBSB  openNewManpage
                SmeBSB  showVersion
                SmeBSB  quit
```

APPLICATION RESOURCES

xman has the following application-specific resources which allow customizations unique to *xman*.

manualFontNormal (Class **Font**)
The font to use for normal text in the manual pages.

manualFontBold (Class **Font**)
The font to use for bold text in the manual pages.

manualFontItalic (Class **Font**)
The font to use for italic text in the manual pages.

directoryFontNormal (Class **Font**)
The font to use for the directory text.

bothShown (Class **Boolean**)
Either ‘true’ or ‘false,’ specifies whether or not you want both the directory and the manual page shown at start up.

directoryHeight (Class **DirectoryHeight**)
The height in pixels of the directory, when the directory and the manual page are shown simultaneously.

topCursor (Class **Cursor**)
The cursor to use in the top box.

helpCursor (Class **Cursor**)
The cursor to use in the help window.

manpageCursor (Class **Cursor**)
The cursor to use in the manual page window.

searchEntryCursor (Class **Cursor**)
The cursor to use in the search entry text widget.

pointerColor (Class **Foreground**)
This is the color of all the cursors (pointers) specified above. The name was chosen to be compatible with xterm.

helpFile (Class **File**) Use this rather than the system default helpfile.

topBox (Class **Boolean**)
Either ‘true’ or ‘false,’ determines whether the top box (containing the help, quit and manual page buttons) or a manual page is put on the screen at start-up. The default is true.

verticalList (Class **Boolean**)
Either ‘true’ or ‘false,’ determines whether the directory listing is vertically or horizontally organized. The default is horizontal (false).

GLOBAL ACTIONS

Xman defines all user interaction through global actions. This allows the user to modify the translation table of any widget, and bind any event to the new user action. The list of actions supported by *xman* are:

GotoPage(*page***)** When used in a manual page display window this will allow the user to move between a directory and manual page display. The *page* argument can be either **Directory** or **ManualPage**.

Quit() This action may be used anywhere, and will exit xman.

Search(*type***,** *action***)** Only useful when used in a search popup, this action will cause the search widget to perform the named search type on the string in the search popup's value widget. This action will also pop down the search widget. The *type* argument can be either **Apropos**, **Manpage** or **Cancel**. If an *action* of **Open** is specified then xman will open a new manual page to display the results of the search, otherwise xman will attempt to display the results in the parent of the search popup.

PopupHelp() This action may be used anywhere, and will popup the help widget.

PopupSearch() This action may be used anywhere except in a help window. It will cause the search popup to become active and visible on the screen, allowing the user search for a manual page.

CreateNewManpage() This action may be used anywhere, and will create a new manual page display window.

RemoveThisManpage() This action may be used in any manual page or help display window. When called it will remove the window, and clean up all resources associated with it.

SaveFormattedPage(*action***)**
This action can only be used in the **likeToSave** popup widget, and tells xman whether to **Save** or **Cancel** a save of the manual page that has just been formatted.

ShowVersion() This action may be called from any manual page or help display window, and will cause the informational display line to show the current version of xman.

FILES

<*manpath directory*>/man<*character*>

<*manpath directory*>/cat<*character*>

<*manpath directory*>/mandesc

/usr/lib/X11/app-defaults/Xman — specifies required resources

/tmp — *Xman* creates temporary files in /tmp for all unformatted man pages and all apropos searches.

SEE ALSO

X(1), *man*(1), *apropos*(1), *catman*(8), *Athena Widget Set*

ENVIRONMENT

DISPLAY — the default host and display to use.

MANPATH — the search path for manual pages. Directories are separated by colons (e.g. /usr/man:/mit/kit/man:/foo/bar/man).

XENVIRONMENT — to get the name of a resource file that overrides the global resources stored in the RESOURCE_MANAGER property.

XAPPLRESDIR — A string that will have ‘‘Xman’’ appended to it. This string will be the full path name of a user app-defaults file to be merged into the resource database after the system app-defaults file, and before the resources that are attached to the display.

COPYRIGHT

AUTHORS

Chris Peterson, MIT X Consortium from the V10 version written by Barry Shein formerly of Boston University.

NAME

xmh – send and read mail with an X interface to MH

SYNOPSIS

xmh [–path *mailpath*] [–initial *foldername*] [–flag] [–*toolkitoption* ...]

DESCRIPTION

The *xmh* program provides a graphical user interface to the *MH* Message Handling System. To actually do things with your mail, it makes calls to the *MH* package. Electronic mail messages may be composed, sent, received, replied to, forwarded, sorted, and stored in folders. *xmh* provides extensive mechanism for customization of the user interface.

This document introduces many aspects of the Athena Widget Set.

OPTIONS

–path *directory*

This option specifies an alternate collection of mail folders in which to process mail. The directory is specified as an absolute pathname. The default mail path is the value of the Path component in the *MH* profile, which is determined by the **MH** environment variable and defaults to $HOME/.mh_profile. $HOME/Mail will be used as the path if the *MH* Path is not given in the profile.

–initial *folder*

This option specifies an alternate folder which may receive new mail and is initially opened by *xmh*. The default initial folder is ‘‘inbox’’.

–flag This option will cause *xmh* to change the appearance of appropriate folder buttons and to request the window manager to change the appearance of the *xmh* icon when new mail has arrived. By default, *xmh* will change the appearance of the ‘‘inbox’’ folder button when new mail is waiting. The application-specific resource **checkNewMail** can be used to turn off this notification, and the **–flag** option will still override it.

These three options have corresponding application-specific resources, **MailPath**, **InitialFolder**, and **MailWaitingFlag**, which can be specified in a resource file.

The standard toolkit command line options are given in *X(1)*.

INSTALLATION

xmh requires that the user is already set up to use *MH*, version 6. To do so, see if there is a file called .mh_profile in your home directory. If it exists, check to see if it contains a line that starts with ‘‘Current-Folder’’. If it does, you’ve been using version 4 or earlier of *MH*; to convert to version 6, you must remove that line. (Failure to do so causes spurious output to stderr, which can hang *xmh* depending on your setup.)

If you do not already have a .mh_profile, you can create one (and everything else you need) by typing ‘‘inc’’ to the shell. You should do this before using *xmh* to incorporate new mail.

For more information, refer to the *mh(1)* documentation.

Much of the user interface of *xmh* is configured in the *Xmh* application class defaults file; if this file was not installed properly a warning message will appear when *xmh* is used. *xmh* is backwards compatible with the R4 application class defaults file.

The default value of the SendBreakWidth resource has changed since R4.

BASIC SCREEN LAYOUT

xmh starts out with a single window, divided into four major areas:

– Six buttons with pull-down command menus.

- A collection of buttons, one for each top level folder. New users of *MH* will have two folders, ‘‘drafts’’ and ‘‘inbox’’.
- A listing, or Table of Contents, of the messages in the open folder. Initially, this will show the messages in ‘‘inbox’’.
- A view of one of your messages. Initially this is blank.

XMH AND THE ATHENA WIDGET SET

xmh uses the X Toolkit Intrinsics and the Athena Widget Set. Many of the features described below (scrollbars, buttonboxes, etc.) are actually part of the Athena Widget Set, and are described here only for completeness. For more information, see the Athena Widget Set documentation.

SCROLLBARS

Some parts of the main window will have a vertical area on the left containing a grey bar. This area is a *scrollbar*. They are used whenever the data in a window takes up more space than can be displayed. The grey bar indicates what portion of your data is visible. Thus, if the entire length of the area is grey, then you are looking at all your data. If only the first half is grey, then you are looking at the top half of your data. The message viewing area will have a horizontal scrollbar if the text of the message is wider than the viewing area.

You can use the pointer in the scrollbar to change what part of the data is visible. If you click with pointer button 2, the top of the grey area will move to where the pointer is, and the corresponding portion of data will be displayed. If you hold down pointer button 2, you can drag around the grey area. This makes it easy to get to the top of the data: just press with button 2, drag off the top of the scrollbar, and release.

If you click with button 1, then the data to the right of the pointer will scroll to the top of the window. If you click with pointer button 3, then the data at the top of the window will scroll down to where the pointer is.

BUTTONBOXES, BUTTONS, AND MENUS

Any area containing many words or short phrases, each enclosed in a rectangular or rounded boundary, is called a *buttonbox*. Each rectangle or rounded area is actually a button that you can press by moving the pointer onto it and pressing pointer button 1. If a given buttonbox has more buttons in it than can fit, it will be displayed with a scrollbar, so you can always scroll to the button you want.

Some buttons have pull-down menus. Pressing the pointer button while the pointer is over one of these buttons will pull down a menu. Continuing to hold the button down while moving the pointer over the menu, called dragging the pointer, will highlight each selectable item on the menu as the pointer passes over it. To select an item in the menu, release the pointer button while the item is highlighted.

ADJUSTING THE RELATIVE SIZES OF AREAS

If you’re not satisfied with the sizes of the various areas of the main window, they can easily be changed. Near the right edge of the border between each region is a black box, called a *grip*. Simply point to that grip with the pointer, press a pointer button, drag up or down, and release. Exactly what happens depends on which pointer button you press.

If you drag with the pointer button 2, then only that border will move. This mode is simplest to understand, but is the least useful.

If you drag with pointer button 1, then you are adjusting the size of the window above. *xmh* will attempt to compensate by adjusting some window below it.

If you drag with pointer button 3, then you are adjusting the size of the window below. *xmh* will attempt to compensate by adjusting some window above it.

All windows have a minimum and maximum size; you will never be allowed to move a border past the point where it would make a window have an invalid size.

PROCESSING YOUR MAIL

This section will define the concepts of the selected folder, current folder, selected message(s), current message, selected sequence, and current sequence. Each *xmh* command is introduced.

For use in customization, action procedures corresponding to each command are given; these action procedures can be used to customize the user interface, particularly the keyboard accelerators and the functionality of the buttons in the optional button box created by the application resource **CommandButtonCount**.

FOLDERS AND SEQUENCES

A folder contains a collection of mail messages, or is empty. *xmh* supports folders with one level of subfolders.

The selected folder is whichever foldername appears in the bar above the folder buttons. Note that this is not necessarily the same folder that is currently being viewed. To change the selected folder, just press on the desired folder button with pointer button 1; if that folder has subfolders, select a folder from the pulldown menu.

The Table of Contents, or toc, lists the messages in the viewed folder. The title bar above the Table of Contents displays the name of the viewed folder.

The toc title bar also displays the name of the viewed sequence of messages within the viewed folder. Every folder has an implicit ‘‘all’’ sequence, which contains all the messages in the folder, and initially the toc title bar will show ‘‘inbox:all’’.

FOLDER COMMANDS

The *Folder* command menu contains commands of a global nature:

Open Folder

Display the data in the selected folder. Thus, the selected folder also becomes the viewed folder. The action procedure corresponding to this command is **XmhOpenFolder(**[*foldername*]**)**. It takes an optional argument as the name of a folder to select and open; if no folder is specified, the selected folder is opened. It may be specified as part of an event translation from a folder menu button or from a folder menu, or as a binding of a keyboard accelerator to any widget other than the folder menu buttons or the folder menus.

Open Folder in New Window

Displays the selected folder in an additional main window. Note, however, that you cannot reliably display the same folder in more than one window at a time, although *xmh* will not prevent you from trying. The corresponding action is **XmhOpenFolderInNewWindow()**.

Create Folder

Create a new folder. You will be prompted for a name for the new folder; to enter the name, move the pointer to the blank box provided and type. Subfolders are created by specifying the parent folder, a slash, and the subfolder name. For example, to create a folder named ‘‘xmh’’ which is a subfolder of an existing folder named ‘‘clients’’, type ‘‘clients/xmh’’. Click on the Okay button when finished, or just type Return; click on Cancel to cancel this operation. The action corresponding to Create Folder is **XmhCreateFolder()**.

Delete Folder

Destroy the selected folder. You will be asked to confirm this action (see CONFIRMATION WINDOWS). Destroying a folder will also destroy any subfolders of that folder. The corresponding action is **XmhDeleteFolder()**.

Close Window
Exits *xmh*, after first confirming that you won't lose any changes; or, if selected from any additional *xmh* window, simply closes that window. The corresponding action is **XmhClose()**.

HIGHLIGHTED MESSAGES, SELECTED MESSAGES AND THE CURRENT MESSAGE

It is possible to highlight a set of adjacent messages in the area of the Table of Contents. To highlight a message, click on it with pointer button 1. To highlight a range of messages, click on the first one with pointer button 1 and on the last one with pointer button 3; or press pointer button 1, drag, and release. To extend a range of selected messages, use pointer button 3. To highlight all messages in the table of contents, click rapidly three times with pointer button 1. To cancel any selection in the table of contents, click rapidly twice.

The selected messages are the same as the highlighted messages, if any. If no messages are highlighted, then the selected messages are considered the same as the current message.

The current message is indicated by a '+' next to the message number. It usually corresponds to the message currently being viewed. Upon opening a new folder, for example, the current message will be different from the viewed message. When a message is viewed, the title bar above the view will identify the message.

TABLE OF CONTENTS COMMANDS

The *Table of Contents* command menu contains commands which operate on the open, or viewed, folder.

Incorporate New Mail
Add any new mail received to viewed folder, and set the current message to be the first new message. This command is selectable in the menu and will execute only if the viewed folder is allowed to receive new mail. By default, only ''inbox'' is allowed to incorporate new mail. The corresponding action is **XmhIncorporateNewMail()**.

Commit Changes
Execute all deletions, moves, and copies that have been marked in this folder. The corresponding action is **XmhCommitChanges()**.

Pack Folder
Renumber the messages in this folder so they start with 1 and increment by 1. The corresponding action is **XmhPackFolder()**.

Sort Folder
Sort the messages in this folder in chronological order. (As a side effect, this may also pack the folder.) The corresponding action is **XmhSortFolder()**.

Rescan Folder
Rebuild the list of messages. This can be used whenever you suspect that *xmh*'s idea of what messages you have is wrong. (In particular, this is necessary if you change things using straight *MH* commands without using *xmh*.) The corresponding action is **XmhForceRescan()**.

MESSAGE COMMANDS

The *Message* command menu contains commands which operate on the selected message(s), or if there are no selected messages, the current message.

Compose Message
Composes a new message. A new window will be brought up for composition; a description of it is given in the COMPOSITION WINDOWS section below. This command does not affect the current message. The corresponding action is **XmhComposeMessage()**.

View Next Message View the first selected message. If no messages are highlighted, view the current message. If current message is already being viewed, view the first unmarked message after the current message. The corresponding action is **XmhViewNextMessage()**.

View Previous View the last selected message. If no messages are highlighted, view the current message. If current message is already being viewed, view the first unmarked message before the current message. The corresponding action is **XmhViewPrevious()**.

Delete Mark the selected messages for deletion. If no messages are highlighted, mark the current message for deletion and automatically display the next unmarked message. The corresponding action is **XmhMarkDeleted()**.

Move Mark the selected messages to be moved into the currently selected folder. (If the selected folder is the same as the viewed folder, this command will just beep.) If no messages are highlighted, mark the current message to be moved and display the next unmarked message. The corresponding action is **XmhMarkMove()**.

Copy as Link Mark the selected messages to be copied into the selected folder. (If the selected folder is the same as the viewed folder, this command will just beep.) If no messages are highlighted, mark the current message to be copied. Note that messages are actually linked, not copied; editing a message copied by *xmh* will affect all copies of the message. The corresponding action is **XmhMarkCopy()**.

Unmark Remove any of the above three marks from the selected messages, or the current message, if none are highlighted. The corresponding action is **XmhUnmark()**.

View in New Create a new window containing only a view of the first selected message, or the current message, if none are highlighted. The corresponding action is **XmhViewInNewWindow()**.

Reply Create a composition window in reply to the first selected message, or the current message, if none are highlighted. The corresponding action is **XmhReply()**.

Forward Create a composition window whose body is initialized to contain an encapsulation of of the selected messages, or the current message if none are highlighted. The corresponding action is **XmhForward()**.

Use as Composition Create a composition window whose body is initialized to be the contents of the first selected message, or the current message if none are selected. Any changes you make in the composition will be saved in a new message in the ‘‘drafts’’ folder, and will not change the original message. However, there is an exception to this rule. If the message to be used as composition was selected from the ‘‘drafts’’ folder, (see BUGS), the changes will be reflected in the original message (see COMPOSITION WINDOWS). The action procedure corresponding to this command is **XmhUseAsComposition()**.

Print Print the selected messages, or the current message if none are selected. *xmh* normally prints by invoking the *enscript*(1) command, but this can be customized with the *xmh* application-specific resource **PrintCommand**. The corresponding action is **XmhPrint()**.

SEQUENCE COMMANDS

The *Sequence* command menu contains commands pertaining to message sequences (See MESSAGE-SEQUENCES), and a list of the message-sequences defined for the currently viewed folder. The selected message-sequence is indicated by a check mark in its entry in the margin of the menu. To change the selected message-sequence, select a new message-sequence from the sequence menu.

Pick Messages Define a new message-sequence. The corresponding action is **XmhPickMessages()**.

The following menu entries will be sensitive only if the current folder has any message-sequences other than the ‘‘all’’ message-sequence.

Open Sequence Change the viewed sequence to be the same as the selected sequence. The corresponding action is **XmhOpenSequence()**.

Add to Sequence Add the selected messages to the selected sequence. The corresponding action is **XmhAddToSequence()**.

Remove from Sequence
Remove the selected messages from the selected sequence. The corresponding action is **XmhRemoveFromSequence()**.

Delete Sequence Remove the selected sequence entirely. The messages themselves are not affected; they simply are no longer grouped together to define a message-sequence. The corresponding action is **XmhDeleteSequence()**.

VIEW COMMANDS

Commands in the *View* menu and in the buttonboxes of view windows (which result from the *Message* menu command **View In New**) correspond in functionality to commands of the same name in the *Message* menu, but they operate on the viewed message rather than the selected messages or current message.

Close Window When the viewed message is in a separate view window, this command will close the view, after confirming the status of any unsaved edits. The corresponding action procedure is **XmhCloseView()**.

Reply Create a composition window in reply to the viewed message. The related action procedure is **XmhViewReply()**.

Forward Create a composition window whose body is initialized contain an encapsulation of the viewed message. The corresponding action is **XmhViewForward()**.

Use As Composition Create a composition window whose body is initialized to be the contents of the viewed message. Any changes made in the composition window will be saved in a new message in the ‘‘drafts’’ folder, and will not change the original message. An exception: if the viewed message was selected from the ‘‘drafts’’ folder, (see BUGS) the original message is edited. The action procedure corresponding to this command is **XmhViewUseAsComposition()**.

Edit Message This command enables the direct editing of the viewed message. The action procedure is **XmhEditView()**.

Save Message This command is insensitive until the message has been edited; when activated, edits will be saved to the original message in the view. The corresponding action is **XmhSaveView()**.

Print Print the viewed message. *xmh* prints by invoking the *enscript*(1) command, but this can be customized with the application-specific resource **PrintCommand**. The corresponding action procedure is **XmhPrintView()**.

Delete Marks the viewed message for deletion. The corresponding action procedure is **XmhViewMarkDelete()**.

OPTIONS

The *Options* menu contains one entry.

Read in Reverse
When selected, a check mark appears in the margin of this menu entry. Read in Reverse will

switch the meaning of the next and previous messages, and will increment to the current message marker in the opposite direction. This is useful if you want to read your messages in the order of most recent first. The option acts as a toggle; select it from the menu a second time to undo the effect. The check mark appears when the option is selected.

COMPOSITION WINDOWS

Composition windows are created by selecting **Compose Message** from the *Message* command menu, or by selecting **Reply** or **Forward** or **Use as Composition** from the *Message* or *View* command menu. These are used to compose mail messages. Aside from the normal text editing functions, there are six command buttons associated with composition windows:

Close Window Close this composition window. If changes have been made since the most recent Save or Send, you will be asked to confirm losing them. The corresponding action is **XmhCloseView()**.

Send Send this composition. The corresponding action is **XmhSend()**.

New Headers Replace the current composition with an empty message. If changes have been made since the most recent Send or Save, you will be asked to confirm losing them. The corresponding action is **XmhResetCompose()**.

Compose Message Bring up another new composition window. The corresponding action is **XmhComposeMessage()**.

Save Message Save this composition in your drafts folder. Then you can safely close the composition. At some future date, you can continue working on the composition by opening the drafts folder, selecting the message, and using the ‘‘Use as Composition’’ command. The corresponding action is **XmhSave()**.

Insert Insert a related message into the composition. If the composition window was created with a ‘‘Reply’’ command, the related message is the message being replied to, otherwise no related message is defined and this button is insensitive. The message may be filtered before being inserted; see **ReplyInsertFilter** under APPLICATION RESOURCES for more information. The corresponding action is **XmhInsert()**.

ACCELERATORS

Accelerators are shortcuts. They allow you to invoke commands without using the menus, either from the keyboard or by using the pointer.

xmh defines pointer accelerators for common actions: To select and view a message with a single click, use pointer button 2 on the message’s entry in the table of contents. To select and open a folder or a sequence in a single action, make the folder or sequence selection with pointer button 2.

To mark the highlighted messages, or current message if none have been highlighted, to be moved to a folder in a single action, use pointer button 3 to select the target folder and simultaneously mark the messages. Similarly, selecting a sequence with pointer button 3 will add the highlighted or current message(s) to that sequence. In both of these operations, the selected folder or sequence and the viewed folder or sequence are not changed.

xmh defines the following keyboard accelerators over the surface of the main window, except in the view area while editing a message:

Meta-I	Incorporate New Mail
Meta-C	Commit Changes
Meta-R	Rescan Folder
Meta-P	Pack Folder
Meta-S	Sort Folder

Meta-space	View Next Message
Meta-c	Mark Copy
Meta-d	Mark Deleted
Meta-f	Forward the selected or current message
Meta-m	Mark Move
Meta-n	View Next Message
Meta-p	View Previous Message
Meta-r	Reply to the selected or current message
Meta-u	Unmark
Ctrl-V	Scroll the table of contents forward
Meta-V	Scroll the table of contents backward
Ctrl-v	Scroll the view forward
Meta-v	Scroll the view backward

TEXT EDITING COMMANDS

All of the text editing commands are actually defined by the Text widget in the Athena Widget Set. The commands may be bound to different keys than the defaults described below through the X Toolkit Intrinsics key re-binding mechanisms. See the X Toolkit Intrinsics and the Athena Widget Set documentation for more details.

Whenever you are asked to enter any text, you will be using a standard text editing interface. Various control and meta keystroke combinations are bound to a somewhat Emacs-like set of commands. In addition, the pointer buttons may be used to select a portion of text or to move the insertion point in the text. Pressing pointer button 1 causes the insertion point to move to the pointer. Double-clicking button 1 selects a word, triple-clicking selects a line, quadruple-clicking selects a paragraph, and clicking rapidly five times selects everything. Any selection may be extended in either direction by using pointer button 3.

In the following, a *line* refers to one displayed row of characters in the window. A *paragraph* refers to the text between carriage returns. Text within a paragraph is broken into lines for display based on the current width of the window. When a message is sent, text is broken into lines based upon the values of the **SendBreakWidth** and **SendWidth** application-specific resources.

The following keystroke combinations are defined:

Ctrl-a	Beginning Of Line	Meta-b	Backward Word
Ctrl-b	Backward Character	Meta-f	Forward Word
Ctrl-d	Delete Next Character	Meta-i	Insert File
Ctrl-e	End Of Line	Meta-k	Kill To End Of Paragraph
Ctrl-f	Forward Character	Meta-q	Form Paragraph
Ctrl-g	Multiply Reset	Meta-v	Previous Page
Ctrl-h	Delete Previous Character	Meta-y	Insert Current Selection
Ctrl-j	Newline And Indent	Meta-z	Scroll One Line Down
Ctrl-k	Kill To End Of Line	Meta-d	Delete Next Word
Ctrl-l	Redraw Display	Meta-D	Kill Word
Ctrl-m	Newline	Meta-h	Delete Previous Word
Ctrl-n	Next Line	Meta-H	Backward Kill Word
Ctrl-o	Newline And Backup	Meta-<	Beginning Of File
Ctrl-p	Previous Line	Meta->	End Of File
Ctrl-r	Search/Replace Backward	Meta-]	Forward Paragraph
Ctrl-s	Search/Replace Forward	Meta-[	Backward Paragraph
Ctrl-t	Transpose Characters		
Ctrl-u	Multiply by 4	Meta-Delete	Delete Previous Word

Ctrl-v	Next Page	Meta-Shift Delete	Kill Previous Word
Ctrl-w	Kill Selection	Meta-Backspace	Delete Previous Word
Ctrl-y	Unkill	Meta-Shift Backspace	Kill Previous Word
Ctrl-z	Scroll One Line Up		

In addition, the pointer may be used to copy and paste text:

Button 1 Down	Start Selection
Button 1 Motion	Adjust Selection
Button 1 Up	End Selection (copy)
Button 2 Down	Insert Current Selection (paste)
Button 3 Down	Extend Current Selection
Button 3 Motion	Adjust Selection
Button 3 Up	End Selection (copy)

CONFIRMATION DIALOG BOXES

Whenever you press a button that may cause you to lose some work or is otherwise dangerous, a popup dialog box will appear asking you to confirm the action. This window will contain an ''Abort'' or ''No'' button and a ''Confirm'' or ''Yes'' button. Pressing the ''No'' button cancels the operation, and pressing the ''Yes'' will proceed with the operation.

Some dialog boxes contain messages from *MH*. Occasionally when the message is more than one line long, not all of the text will be visible. Clicking on the message field will cause the dialog box to resize so that you can read the entire message.

MESSAGE-SEQUENCES

An *MH* message sequence is just a set of messages associated with some name. They are local to a particular folder; two different folders can have sequences with the same name. The sequence named ''all'' is predefined in every folder; it consists of the set of all messages in that folder. As many as nine sequences may be defined for each folder, including the predefined ''all'' sequence. (The sequence ''cur'' is also usually defined for every folder; it consists of only the current message. *xmh* hides ''cur'' from the user, instead placing a ''+'' by the current message. Also, *xmh* does not support *MH*'s''unseen'' sequence, so that one is also hidden from the user.)

The message sequences for a folder (including one for ''all'') are displayed in the ''Sequence'' menu, below the sequence commands. The table of contents (also known as the ''toc'') is at any one time displaying one message sequence. This is called the ''viewed sequence'', and its name will be displayed in the toc title bar after the folder name. Also, at any time one of the sequences in the menu will have a check mark next to it. This is called the ''selected sequence''. Note that the viewed sequence and the selected sequence are not necessarily the same. (This all pretty much corresponds to the way folders work.)

The **Open Sequence**, **Add to Sequence**, **Remove from Sequence**, and **Delete Sequence** commands are active only if the viewed folder contains message-sequences other than ''all'' sequence.

Note that none of the above actually affect whether a message is in the folder. Remember that a sequence is a set of messages within the folder; the above operations just affect what messages are in that set.

To create a new sequence, select the ''Pick'' menu entry. A new window will appear, with lots of places to enter text. Basically, you can describe the sequence's initial set of messages based on characteristics of the message. Thus, you can define a sequence to be all the messages that were from a particular person, or with a particular subject, and so on. You can also connect things up with boolean operators, so you can select all things from ''weissman'' with a subject containing ''xmh''.

The layout should be fairly obvious. The simplest cases are the easiest: just point to the proper field and type. If you enter in more than one field, it will only select messages which match all non-empty fields.

The more complicated cases arise when you want things that match one field or another one, but not necessarily both. That's what all the ''or'' buttons are for. If you want all things with subjects that include ''xmh'' or ''xterm'', just press the ''or'' button next to the ''Subject:'' field. Another box will appear where you can enter another subject.

If you want all things either from ''weissman'' or with subject ''xmh'', but not necessarily both, select the ''–Or–'' button. This will essentially double the size of the form. You can then enter ''weissman'' in a from: box on the top half, and ''xmh'' in a subject: box on the lower part.

If you select the ''Skip'' button, then only those messages that *don't* match the fields on that row are included.

Finally, in the bottom part of the window will appear several more boxes. One is the name of the sequence you're defining. (It defaults to the name of the selected sequence when ''Pick'' was pressed, or to ''temp'' if ''all'' was the selected sequence.) Another box defines which sequence to look through for potential members of this sequence; it defaults to the viewed sequence when ''Pick'' was pressed.

Two more boxes define a date range; only messages within that date range will be considered. These dates must be entered in RFC 822-style format: each date is of the form ''dd mmm yy hh:mm:ss zzz'', where dd is a one or two digit day of the month, mmm is the three-letter abbreviation for a month, and yy is a year. The remaining fields are optional: hh, mm, and ss specify a time of day, and zzz selects a time zone. Note that if the time is left out, it defaults to midnight; thus if you select a range of ''7 nov 86'' – ''8 nov 86'', you will only get messages from the 7th, as all messages on the 8th will have arrived after midnight.

''Date field'' specifies which field in the header to look at for this date range; it defaults to ''Date''. If the sequence you're defining already exists, you can optionally merge the old set with the new; that's what the ''Yes'' and ''No'' buttons are all about. Finally, you can ''OK'' the whole thing, or ''Cancel'' it.

In general, most people will rarely use these features. However, it's nice to occasionally use ''Pick'' to find some messages, look through them, and then hit ''Delete Sequence'' to put things back in their original state.

WIDGET HIERARCHY

In order to specify resources, it is useful to know the hierarchy of widgets which compose *xmh*. In the notation below, indentation indicates hierarchical structure. The widget class name is given first, followed by the widget instance name. The application class name is Xmh.

The hierarchy of the main toc and view window is identical for additional toc and view windows, except that a TopLevelShell widget is inserted in the hierarchy between the application shell and the Paned widget.

```
Xmh xmh
    Paned xmh
        SimpleMenu  folderMenu
            SmeBSB  open
            SmeBSB  openInNew
            SmeBSB  create
            SmeBSB  delete
            SmeLine  line
            SmeBSB  close
        SimpleMenu  tocMenu
            SmeBSB  inc
```

```
                SmeBSB  commit
                SmeBSB  pack
                SmeBSB  sort
                SmeBSB  rescan
        SimpleMenu  messageMenu
                SmeBSB  compose
                SmeBSB  next
                SmeBSB  prev
                SmeBSB  delete
                SmeBSB  move
                SmeBSB  copy
                SmeBSB  unmark
                SmeBSB  viewNew
                SmeBSB  reply
                SmeBSB  forward
                SmeBSB  useAsComp
                SmeBSB  print
        SimpleMenu  sequenceMenu
                SmeBSB  pick
                SmeBSB  openSeq
                SmeBSB  addToSeq
                SmeBSB  removeFromSeq
                SmeBSB  deleteSeq
                SmeLine  line
                SmeBSB  all
        SimpleMenu  viewMenu
                SmeBSB  reply
                SmeBSB  forward
                SmeBSB  useAsComp
                SmeBSB  edit
                SmeBSB  save
                SmeBSB  print
        SimpleMenu  optionMenu
                SmeBSB  reverse
        Viewport.Core  menuBox.clip
                Box  menuBox
                        MenuButton  folderButton
                        MenuButton  tocButton
                        MenuButton  messageButton
                        MenuButton  sequenceButton
                        MenuButton  viewButton
                        MenuButton  optionButton
        Grip  grip
        Label folderTitlebar
        Grip  grip
        Viewport.Core  folders.clip
                Box  folders
                        MenuButton  inbox
                        MenuButton  drafts
                                SimpleMenu  menu
                                        SmeBSB <folder_name>
                                                .
                                                .
```

.

```
        Grip  grip
        Label  tocTitlebar
        Grip  grip
        Text toc
                Scrollbar  vScrollbar
        Grip  grip
        Label  viewTitlebar
        Grip  grip
        Text  view
                Scrollbar  vScrollbar
                Scrollbar  hScrollbar
```

The hierarchy of the Create Folder popup dialog box:

```
TransientShell  prompt
        Dialog  dialog
                Label  label
                Text  value
                Command  okay
                Command  cancel
```

The hierarchy of the Notice dialog box, which reports messages from MH:

```
TransientShell  notice
        Dialog  dialog
                Label  label
                Text  value
                Command  confirm
```

The hierarchy of the Confirmation dialog box:

```
TransientShell  confirm
        Dialog  dialog
                Label  label
                Command  yes
                Command  no
```

The hierarchy of the dialog box which reports errors:

```
TransientShell  error
        Dialog  dialog
                Label  label
                Command  OK
```

The hierarchy of the composition window:

```
TopLevelShell  xmh
        Paned  xmh
                Label  composeTitlebar
                Text  comp
                Viewport.Core  compButtons.clip
```

```
                Box  compButtons
                     Command  close
                     Command  send
                     Command  reset
                     Command  compose
                     Command  save
                     Command  insert
```

The hierarchy of the view window:

```
TopLevelShell  xmh
     Paned  xmh
          Label  viewTitlebar
          Text  view
          Viewport.Core  viewButtons.clip
               Box  viewButtons
                    Command  close
                    Command  reply
                    Command  forward
                    Command  useAsComp
                    Command  edit
                    Command  save
                    Command  print
                    Command  delete
```

The hierarchy of the pick window:
(Unnamed widgets have no name.)

```
TopLevelShell  xmh
     Paned  xmh
          Label  pickTitlebar
          Viewport.Core  pick.clip
               Form  form
                    Form  groupform
```

The first 6 rows of the pick window have identical structure:

```
                         Form  rowform
                              Toggle
                              Toggle
                              Label
                              Text
                              Command

                         Form  rowform
                              Toggle
                              Toggle
                              Text
                              Text
                              Command
                         Form  rowform
                              Command
          Viewport.core  pick.clip
               Form  form
                    From  groupform
```

```
Form  rowform
        Label
        Text
        Label
        Text
Form  rowform
        Label
        Text
        Label
        Text
        Label
        Text
Form  rowform
        Label
        Toggle
        Toggle
Form  rowform
        Command
        Command
```

APPLICATION-SPECIFIC RESOURCES

The application class name is **Xmh**. Application-specific resources are listed below by name. Application-specific resource class names always begin with an upper case character, but unless noted, are otherwise identical to the instance names given below.

Any of these options may also be specified on the command line by using the X Toolkit Intrinsics resource specification mechanism. Thus, to run *xmh* showing all message headers,
% xmh –xrm '*HideBoringHeaders:off'

If **TocGeometry**, **ViewGeometry**, **CompGeometry**, or **PickGeometry** are not specified, then the value of **Geometry** is used instead. If the resulting height is not specified (e.g., "", "=500", "+0-0"), then the default height of windows is calculated from fonts and line counts. If the width is not specified (e.g., "", "=x300", "-0+0"), then half of the display width is used. If unspecified, the height of a pick window defaults to half the height of the display.

The following resources are defined:

banner A short string that is the default label of the folder, Table of Contents, and view. The default is "xmh MIT X Consortium R5".

blockEventsOnBusy
Whether to disallow user input and show a busy cursor while *xmh* is busy processing a command. Default is true.

busyCursor
The name of the symbol used to represent the position of the pointer, displayed if **blockEventsOnBusy** is true, when *xmh* is processing a time-consuming command. The default is "watch".

busyPointerColor
The foreground color of the busy cursor. Default is XtDefaultForeground.

checkFrequency
How often to check for new mail, make checkpoints, and rescan the Table of Contents, in minutes. If **checkNewMail** is true, *xmh* checks to see if you have new mail each interval. If **makeCheckpoints** is true, checkpoints are made every fifth interval. Also every fifth interval, the Table of Contents is checked for inconsistencies with the file system, and rescanned if out of date. To prevent all of these checks from occurring, set **CheckFrequency** to 0. The default is 1.

This resource is retained for backward compatibility with user resource files; see also **checkpointInterval**, **mailInterval**, and **rescanInterval**.

checkNewMail
If true, *xmh* will check at regular intervals to see if new mail has arrived for any of the top level folders and any opened subfolders. A visual indication will be given if new mail is waiting to be incorporated into a top level folder. Default is true. The interval can be adjusted with **mailInterval**.

checkpointInterval (class **Interval**)
Specifies in minutes how often to make checkpoints of volatile state, if **makeCheckpoints** is true. The default is 5 times the value of **checkFrequency**.

checkpointNameFormat
Specifies how checkpointed files are to be named. The value of this resource will be used to compose a file name by inserting the message number as a string in place of the required single occurance of '%d'. If the value of the resource is the empty string, or if no '%d' occurs in the string, or if "%d" is the value of the resource, the default will be used instead. The default is "%d.CKP". Checkpointing is done in the folder of origin unless an absolute pathname is given. *xmh* does not assist the user in recovering checkpoints, nor does it provide for removal of the checkpoint files.

commandButtonCount
The number of command buttons to create in a button box in between the toc and the view areas of the main window. *xmh* will create these buttons with the names *button1*, *button2* and so on, in a box with the name *commandBox*. The default is 0. *xmh* users can specify labels and actions for the buttons in a private resource file; see the section ACTIONS AND INTERFACE CUSTOMIZATION.

compGeometry
Initial geometry for windows containing compositions.

cursor The name of the symbol used to represent the pointer. Default is ''left_ptr''.

debug Whether or not to print information to stderr as *xmh* runs. Default is false.

draftsFolder
The folder used for message drafts. Default is ''drafts''.

geometry
Default geometry to use. Default is none.

hideBoringHeaders
If ''on'', then *xmh* will attempt to skip uninteresting header lines within messages by scrolling them off the top of the view. Default is ''on''.

initialFolder
Which folder to display on startup. May also be set with the command-line option **–initial**. Default is ''inbox''.

initialIncFile
The absolute path name of your incoming mail drop file. In some installations, for example those using the Post Office Protocol, no file is appropriate. In this case, **initialIncFile** should not be specified, or may be specified as the empty string, and *inc* will be invoked without a –file argument. By default, this resource has no value. This resource is ignored if *xmh* finds an *.xmhcheck* file; see the section on multiple mail drops.

mailInterval (class **Interval)**
Specifies the interval in minutes at which the mail should be checked, if **mailWaitingFlag** or **checkNewMail** is true. The default is the value of **checkFrequency**.

mailPath
The full path prefix for locating your mail folders. May also be set with the command line option, **–path**. The default is the Path component in the *MH* profile, or ‘‘$HOME/Mail’’ if none.

mailWaitingFlag
If true, *xmh* will attempt to set an indication in its icon when new mail is waiting to be retrieved. If **mailWaitingFlag** is true, then **checkNewMail** is assumed to be true as well. The **–flag** command line option is a quick way to turn on this resource.

makeCheckpoints
If true, *xmh* will attempt to save checkpoints of volatile edits. The default is false. The frequency of checkpointing is controlled by the resource **checkpointInterval**. For the location of checkpointing, see **checkpointNameFormat**.

mhPath What directory in which to find the *MH* commands. If a command isn’t found in the user’s path, then the path specified here is used. Default is ‘‘/usr/local/mh6’’.

newMailBitmap (class **NewMailBitmap**)
The bitmap to show in the folder button when a folder has new mail. The default is ‘‘black6’’.

newMailIconBitmap (class **NewMailBitmap**)
The bitmap suggested to the window manager for the icon when any folder has new mail. The default is ‘‘flagup’’.

noMailBitmap (class **NoMailBitmap)**
The bitmap to show in the folder button when a folder has no new mail. The default is ‘‘box6’’.

noMailIconBitmap (class **NoMailBitmap)**
The bitmap suggested to the window manager for the icon when no folders have new mail. The default is ‘‘flagdown’’.

pickGeometry
Initial geometry for pick windows.

pointerColor
The foreground color of the pointer. Default is XtDefaultForeground.

prefixWmAndIconName
Whether to prefix the window and icon name with "xmh: ". Default is true.

printCommand
An *sh* command to execute to print a message. Note that stdout and stderr must be specifically redirected. If a message or range of messages is selected for printing, the full file paths of each message file are appended to the specified print command. The default is ‘‘enscript >/dev/null 2>/dev/null’’.

replyInsertFilter
An *sh* command to be executed when the *Insert* button is activated in a composition window. The full path and filename of the source message is appended to the command before being passed to *sh*(1). The default filter is *cat*; i.e. it inserts the entire message into the composition. Interesting filters are: *sed 's/^/> /'* or *awk -e '{print "   " $0}'* or *<mh directory>/lib/mhl –form mhl.body*.

rescanInterval (class **Interval**)
How often to check the Table of Contents of currently viewed folders and of folders with messages currently being viewed, and to update the Table of Contents if *xmh* sees inconsistencies with the file system in these folders. The default is 5 times the value of **checkFrequency**.

reverseReadOrder
When true, the next message will be the message prior to the current message in the table of contents, and the previous message will be the message after the current message in the table of contents. The default is false.

sendBreakWidth
When a message is sent from *xmh*, lines longer than this value will be split into multiple lines, each of which is no longer than **SendWidth**. This value may be overridden for a single message by inserting an additional line in the message header of the form *SendBreakWidth: value*. This line will be removed from the header before the message is sent. The default is 2000 (to allow for sending mail containing source patches).

sendWidth
When a message is sent from *xmh*, lines longer than **SendBreakWidth** characters will be split into multiple lines, each of which is no longer than this value. This value may be overridden for a single message by inserting an additional line in the message header of the form *SendWidth: value*. This line will be removed from the header before the message is sent. The default is 72.

showOnInc
Whether to automatically show the current message after incorporating new mail. Default is true.

skipCopied
Whether to skip over messages marked for copying when using ‘‘View Next Message’’ and ‘‘View Previous Message’’. Default is true.

skipDeleted
Whether to skip over messages marked for deletion when using ‘‘View Next Message’’ and ‘‘View Previous Message’’. Default is true.

skipMoved
Whether to skip over messages marked for moving to other folders when using ‘‘View Next Message’’ and ‘‘View Previous Message’’. Default is true.

stickyMenu
If true, when popup command menus are used, the most recently selected entry will be under the cursor when the menu pops up. Default is false. See the file *clients/xmh/Xmh.sample* for an example of how to specify resources for popup command menus.

tempDir Directory for *xmh* to store temporary files. For privacy, a user might want to change this to a private directory. Default is ‘‘/tmp’’.

tocGeometry
Initial geometry for main *xmh* toc and view windows.

tocPercentage
The percentage of the main window that is used to display the Table of Contents. Default is 33.

tocWidth
How many characters to generate for each message in a folder’s table of contents. Default is 100. Use less if the geometry of the main *xmh* window results in the listing being clipped at the right hand boundary, or if you plan to use *mhl* a lot, because it will be faster, and the extra characters may not be useful.

viewGeometry
Initial geometry for windows showing a view of a message.

MULTIPLE MAIL DROPS

Users may need to incorporate mail from multiple spool files or mail drops. If incoming mail is forwarded to the *MH slocal* program, it can be sorted as specified by the user into multiple incoming mail drops. Refer to the *MH* man page for *slocal* to learn how to specify fowarding and the automatic sorting of incoming mail in a *.maildelivery* file.

To inform *xmh* about the various mail drops, create a file in your home directory called *.xmhcheck*. In this file, a mapping between existing folder names and mail drops is created by giving a folder name followed by the absolute pathname of the mail drop site, with some white space separating them, one mapping per line. *xmh* will read this file whether or not resources are set for notification of new mail arrival, and will allow incorporation of new mail into any folder with a mail drop. *xmh* will invoke *inc* with the *–file* argument, and if *xmh* has been requested to check for new mail, it will check directly, instead of using *msgchk*.

An example of *.xmhcheck* file format, for the folders ‘‘inbox’’ and ‘‘xpert’’:

```
inbox   /usr/spool/mail/converse
xpert   /users/converse/maildrops/xpert
```

ACTIONS AND INTERFACE CUSTOMIZATION

Because *xmh* provides action procedures which correspond to command functionality and installs accelerators, users can customize accelerators and new button functionality in a private resource file. For examples of specifying customized resources, see the file *mit/clients/xmh/Xmh.sample*. To understand the syntax, see the Appendix of the *X Toolkit Intrinsics* specification on *Translation Table Syntax*, and any general explanation of using and specifying *X* resources. Unpredictable results can occur if actions are bound to events or widgets for which they were not designed.

Here’s an example of how to bind actions to your own *xmh* buttons, and how to redefine the default accelerators so that the Meta key is not required, in case you don’t have access to the sample file mentioned above.

```
! To create buttons in the middle of the main window and give them semantics:

Xmh*CommandButtonCount:         5

Xmh*commandBox.button1.label:  Inc
Xmh*commandBox.button1.translations: #override\
        <Btn1Down>,<Btn1Up>: XmhIncorporateNewMail() unset()

Xmh*commandBox.button2.label:  Compose
Xmh*commandBox.button2.translations: #override\
        <Btn1Down>,<Btn1Up>: XmhComposeMessage() unset()

Xmh*commandBox.button3.label:  Next
Xmh*commandBox.button3.translations: #override\
        <Btn1Down>,<Btn1Up>: XmhViewNextMessage() unset()

Xmh*commandBox.button4.label:  Delete
Xmh*commandBox.button4.translations: #override\
        <Btn1Down>,<Btn1Up>: XmhMarkDelete() unset()

Xmh*commandBox.button5.label:  Commit
Xmh*commandBox.button5.translations: #override\
        <Btn1Down>,<Btn1Up>: XmhCommitChanges() unset()

! To redefine the accelerator bindings to exclude modifier keys,
! and add your own keyboard accelerator for Compose Message:

Xmh*tocMenu.accelerators: #override\n\
        !:<Key>I:       XmhIncorporateNewMail()\n\
        !:<Key>C:       XmhCommitChanges()\n\
        !:<Key>R:       XmhForceRescan()\n\
        !:<Key>P:       XmhPackFolder()\n\
        !:<Key>S:       XmhSortFolder()\n
Xmh*messageMenu.accelerators: #override\n\
```

```
!:<Key>E:       XmhComposeMessage()\n\
!<Key>space:    XmhViewNextMessage()\n\
!:<Key>c:       XmhMarkCopy()\n\
!:<Key>d:       XmhMarkDelete()\n\
!:<Key>f:       XmhForward()\n\
!:<Key>m:       XmhMarkMove()\n\
!:<Key>n:       XmhViewNextMessage()\n\
!:<Key>p:       XmhViewPreviousMessage()\n\
!:<Key>r:       XmhReply()\n\
!:<Key>u:       XmhUnmark()\n
```

xmh provides action procedures which correspond to entries in the command menus; these are given in the sections describing menu commmands, not here. In addition to the actions corresponding to commands in the menus, these action routines are defined:

XmhPushFolder([*foldername, ...*]**)**
This action pushes each of its argument(s) onto a stack of foldernames. If no arguments are given, the selected folder is pushed onto the stack.

XmhPopFolder()
This action pops one foldername from the stack and sets the selected folder.

XmhPopupFolderMenu()
This action should always be taken when the user selects a folder button. A folder button represents a folder and zero or more subfolders. The menu of subfolders is built upon the first reference, by this routine. If there are no subfolders, this routine will mark the folder as having no subfolders, and no menu will be built. In that case the menu button emulates a toggle button. When subfolders exist, the menu will popup, using the menu button action PopupMenu().

XmhSetCurrentFolder()
This action allows menu buttons to emulate toggle buttons in the function of selecting a folder. This action is for menu button widgets only, and sets the selected folder.

XmhLeaveFolderButton()
This action ensures that the menu button behaves properly when the user moves the pointer out of the menu button window.

XmhPushSequence([*sequencename, ...*]**)**
This action pushes each of its arguments onto the stack of sequence names. If no arguments are given, the selected sequence is pushed onto the stack.

XmhPopSequence()
This action pops one sequence name from the stack of sequence names, which then becomes the selected sequence.

XmhPromptOkayAction()
This action is equivalent to pressing the okay button in the Create Folder popup.

XmhReloadSeqLists()
This action rescans the contents of the public *MH* sequences for the currently opened folder and updates the sequence menu if necessary.

XmhShellCommand(*parameter* [, *parameter*]**)**
At least one parameter must be specified. The parameters will be concatenated with a space character separator, into a single string, and the list of selected messsages, or if no messages are selected, the current message, will be appended to the string of parameters. The string will be executed as a shell command. The messages are always given as absolute pathnames. It is an error to cause this action to execute when there are no selected messages and no current message.

XmhCheckForNewMail()

This action will check all mail drops known to xmh. If no mail drops have been specified by the user either through the *.xmhcheck* file or by the **initialIncFile** resource, the *MH* command *msgchk* is used to check for new mail, otherwise, *xmh* checks directly.

XmhWMProtocols([wm_delete_window] [wm_save_yourself])
This action is responsible for participation in window manager communication protocols. It responds to delete window and save yourself messages. The user can cause *xmh* to respond to one or both of these protocols, exactly as if the window manager had made the request, by invoking the action with the appropriate parameters. The action is insensitive to the case of the string parameters. If the event received is a ClientMessage event and parameters are present, at least one of the parameters must correspond to the protocol requested by the event for the request to be honored by *xmh*.

CUSTOMIZATION USING *MH*

The initial text displayed in a composition window is generated by executing the corresponding *MH* command; i.e. *comp*, *repl*, or *forw*, and therefore message components may be customized as specified for those commands. *comp* is executed only once per invocation of *xmh* and the message template is re-used for every successive new composition.

xmh uses *MH* commands, including *inc, msgchk, comp, send, repl, forw, refile, rmm, pick, pack, sort,* and *scan*. Some flags for these commands can be specified in the *MH* profile; *xmh* may override them. The application resource **debug** can be set to true to see how *xmh* uses *MH* commands.

ENVIRONMENT

HOME - users's home directory
MH - to get the location of the *MH* profile file

FILES

˜/.mh_profile - *MH* profile, used if the MH environment variable is not set
˜/Mail - directory of folders, used if the *MH* profile cannot be found
˜/.xmhcheck - optional, for multiple mail drops in cooperation with *slocal*.
/usr/local/mh6 - *MH* commands, as a last resort, see **mhPath**.
˜/Mail/<folder>/.xmhcache - *scan* output in each folder
˜/Mail/<folder>/.mh_sequences - sequence definitions, in each folder
/tmp - temporary files, see **tempDir**.

SEE ALSO

X(1), xrdb(1), X Toolkit Intrinsics, Athena Widget Set, mh(1), enscript(1)
At least one book has been published about *MH* and *xmh*.

BUGS

- When the user closes a window, all windows which are transient for that window should also be closed by *xmh*.
- When **XmhUseAsComposition** and **XmhViewUseAsComposition** operate on messages in the **DraftsFolder**, *xmh* disallows editing of the composition if the same message is also being viewed in another window.
- Occasionally after committing changes, the table of contents will appear to be completely blank when there are actually messages present. When this happens, refreshing the display, or typing Control-L in the table of contents, will often cause the correct listing to appear. If this doesn't work, force a rescan of the folder.
- Should recognize and use the ''unseen'' message-sequence.
- Should determine by itself if the user hasn't used *MH* before, and offer to create the .mh_profile, instead of hanging on inc.
- A few commands are missing (rename folder, resend message).
- WM_DELETE_WINDOW protocol doesn't work right when requesting deletion of the first toc and view, while trying to keep other *xmh* windows around.

- Doesn’t support annotations when replying to messages.
- Doesn’t allow folders to be shared without write permission.
- Doesn’t recognize private sequences.
- *MH* will report that the *.mh_sequences* file is poorly formatted if any sequence definition in a particular folder contains more than *BUFSIZ* characters. *xmh* tries to capture these messages and display them when they occur, but it cannot correct the problem.

COPYRIGHT

AUTHOR

Terry Weissman, formerly of Digital Western Research Laboratory
Donna Converse, MIT X Consortium

NAME

xmkmf – create a Makefile from an Imakefile

SYNOPSIS

xmkmf [-a] [*topdir* [*curdir*]]

DESCRIPTION

The *xmkmf* command is the normal way to create a *Makefile* from an *Imakefile* shipped with third-party software.

When invoked with no arguments in a directory containing an *Imakefile,* the *imake* program is run with arguments appropriate for your system (configured into *xmkmf* when X was built) and generates a *Makefile.*

When invoked with the *–a* option, *xmkmf* builds the *Makefile* in the current directory, and then automatically executes ‘‘make Makefiles’’ (in case there are subdirectories), ‘‘make includes’’, and ‘‘make depend’’ for you. This is the normal way to configure software that is outside the MIT X build tree.

If working inside the MIT X build tree (unlikely unless you are an X developer, and even then this option is never really used), the *topdir* argument should be specified as the relative pathname from the current directory to the top of the build tree. Optionally, *curdir* may be specified as a relative pathname from the top of the build tree to the current directory. It is necessary to supply *curdir* if the current directory has subdirectories, or the *Makefile* will not be able to build the subdirectories. If a *topdir* is given, *xmkmf* assumes nothing is installed on your system and looks for files in the build tree instead of using the installed versions.

SEE ALSO

imake(1)

NAME

xmodmap - utility for modifying keymaps in X

SYNOPSIS

xmodmap [-options ...] [filename]

DESCRIPTION

The *xmodmap* program is used to edit and display the keyboard *modifier map* and *keymap table* that are used by client applications to convert event keycodes into keysyms. It is usually run from the user's session startup script to configure the keyboard according to personal tastes.

OPTIONS

The following options may be used with *xmodmap*:

–display *display*
This option specifies the host and display to use.

–help This option indicates that a brief description of the command line arguments should be printed on the standard error channel. This will be done whenever an unhandled argument is given to *xmodmap.*

–grammar
This option indicates that a help message describing the expression grammar used in files and with -e expressions should be printed on the standard error.

–verbose
This option indicates that *xmodmap* should print logging information as it parses its input.

–quiet This option turns off the verbose logging. This is the default.

–n This option indicates that *xmodmap* should not change the mappings, but should display what it would do, like *make(1)* does when given this option.

–e *expression*
This option specifies an expression to be executed. Any number of expressions may be specified from the command line.

–pm This option indicates that the current modifier map should be printed on the standard output.

–pk This option indicates that the current keymap table should be printed on the standard output.

–pke This option indicates that the current keymap table should be printed on the standard output in the form of expressions that can be fed back to *xmodmap*.

–pp This option indicates that the current pointer map should be printed on the standard output.

– A lone dash means that the standard input should be used as the input file.

The *filename* specifies a file containing *xmodmap* expressions to be executed. This file is usually kept in the user's home directory with a name like *.xmodmaprc*.

EXPRESSION GRAMMAR

The *xmodmap* program reads a list of expressions and parses them all before attempting to execute any of them. This makes it possible to refer to keysyms that are being redefined in a natural way without having to worry as much about name conflicts.

keycode *NUMBER = KEYSYMNAME ...*
The list of keysyms is assigned to the indicated keycode (which may be specified in decimal, hex or octal and can be determined by running the *xev* program in the examples directory).

keysym *KEYSYMNAME = KEYSYMNAME ...*
The *KEYSYMNAME* on the left hand side is translated into matching keycodes used to perform the corresponding set of **keycode** expressions. The list of keysym names may be found in the header file *<X11/keysymdef.h>* (without the *XK_* prefix) or the keysym database */usr/lib/X11/XKeysymDB*. Note that if the same keysym is bound to multiple keys, the expression

is executed for each matching keycode.

clear *MODIFIERNAME*

This removes all entries in the modifier map for the given modifier, where valid name are: **Shift**, **Lock**, **Control**, **Mod1**, **Mod2**, **Mod3**, **Mod4**, and **Mod5** (case does not matter in modifier names, although it does matter for all other names). For example, ‘‘clear Lock’’ will remove all any keys that were bound to the shift lock modifier.

add *MODIFIERNAME = KEYSYMNAME ...*

This adds all keys containing the given keysyms to the indicated modifier map. The keysym names are evaluated after all input expressions are read to make it easy to write expressions to swap keys (see the EXAMPLES section).

remove *MODIFIERNAME = KEYSYMNAME ...*

This removes all keys containing the given keysyms from the indicated modifier map. Unlike **add,** the keysym names are evaluated as the line is read in. This allows you to remove keys from a modifier without having to worry about whether or not they have been reassigned.

pointer = default

This sets the pointer map back to its default settings (button 1 generates a code of 1, button 2 generates a 2, etc.).

pointer = *NUMBER ...*

This sets to pointer map to contain the indicated button codes. The list always starts with the first physical button.

Lines that begin with an exclamation point (!) are taken as comments.

If you want to change the binding of a modifier key, you must also remove it from the appropriate modifier map.

EXAMPLES

Many pointers are designed such that the first button is pressed using the index finger of the right hand. People who are left-handed frequently find that it is more comfortable to reverse the button codes that get generated so that the primary button is pressed using the index finger of the left hand. This could be done on a 3 button pointer as follows:

```
% xmodmap -e "pointer = 3 2 1"
```

Many editor applications support the notion of Meta keys (similar to Control keys except that Meta is held down instead of Control). However, some servers do not have a Meta keysym in the default keymap table, so one needs to be added by hand. The following command will attach Meta to the Multi-language key (sometimes labeled Compose Character). It also takes advantage of the fact that applications that need a Meta key simply need to get the keycode and don’t require the keysym to be in the first column of the keymap table. This means that applications that are looking for a Multi_key (including the default modifier map) won’t notice any change.

```
% xmodmap -e "keysym Multi_key = Multi_key Meta_L"
```

One of the more simple, yet convenient, uses of *xmodmap* is to set the keyboard’s "rubout" key to generate an alternate keysym. This frequently involves exchanging Backspace with Delete to be more comfortable to the user. If the *ttyModes* resource in *xterm* is set as well, all terminal emulator windows will use the same key for erasing characters:

```
% xmodmap -e "keysym BackSpace = Delete"
% echo "XTerm*ttyModes: erase ˆ?" | xrdb -merge
```

Some keyboards do not automatically generate less than and greater than characters when the comma and period keys are shifted. This can be remedied with *xmodmap* by resetting the bindings for the comma and period with the following scripts:

```
!
! make shift-, be < and shift-. be >
!
keysym comma = comma less
keysym period = period greater
```

One of the more irritating differences between keyboards is the location of the Control and Shift Lock keys. A common use of *xmodmap* is to swap these two keys as follows:

```
!
! Swap Caps_Lock and Control_L
!
remove Lock = Caps_Lock
remove Control = Control_L
keysym Control_L = Caps_Lock
keysym Caps_Lock = Control_L
add Lock = Caps_Lock
add Control = Control_L
```

The *keycode* command is useful for assigning the same keysym to multiple keycodes. Although unportable, it also makes it possible to write scripts that can reset the keyboard to a known state. The following script sets the backspace key to generate Delete (as shown above), flushes all existing caps lock bindings, makes the CapsLock key be a control key, make F5 generate Escape, and makes Break/Reset be a shift lock.

```
!
! On the HP, the following keycodes have key caps as listed:
!
!     101  Backspace
!      55  Caps
!      14  Ctrl
!      15  Break/Reset
!      86  Stop
!      89  F5
!
keycode 101 = Delete
keycode 55 = Control_R
clear Lock
add Control = Control_R
keycode 89 = Escape
keycode 15 = Caps_Lock
add Lock = Caps_Lock
```

ENVIRONMENT

DISPLAY

to get default host and display number.

SEE ALSO

X(1), xev(1), *Xlib* documentation on key and pointer events

BUGS

Every time a **keycode** expression is evaluated, the server generates a *MappingNotify* event on every client. This can cause some thrashing. All of the changes should be batched together and done at once. Clients that receive keyboard input and ignore *MappingNotify* events will not notice any changes made to keyboard mappings.

Xmodmap should generate "add" and "remove" expressions automatically whenever a keycode that is already bound to a modifier is changed.

There should be a way to have the *remove* expression accept keycodes as well as keysyms for those times when you really mess up your mappings.

COPYRIGHT

AUTHOR

Jim Fulton, MIT X Consortium, rewritten from an earlier version by David Rosenthal of Sun Microsystems.

NAME

xon – start an X program on a remote machine

SYNOPSIS

xon remote-host [-access] [-debug] [-name window-name] [-nols] [-screen screen-no] [-user user-name] [command ...]

DESCRIPTION

Xon runs the specified command (default xterm -ls) on the remote machine using rsh. Xon passes the DISPLAY, XAUTHORITY and XUSERFILESEARCHPATH environment variables to the remote command.

When no command is specified, xon runs ’xterm -ls’. It additionally specifies the application name to be ’xterm-*remote-host*’ and the window title to be ’-fIremote-host’.

Xon can only work when the remote host will allow you to use rsh, by having an entry in the .rhosts file permitting access.

OPTIONS

Note that the options follow the remote host name (as they do with rlogin).

-access Runs xhost locally to add the remote host to the host access list in the X server. This won’t work unless xhost is given permission to modify the access list.

-debug Normally, xon disconnects the remote process from stdin, stdout and stderr to eliminate the daemon processes which usually connect them across the network. Specifying the **-debug** option leaves them connected so that error messages from the remote execution are sent back to the originating host.

-name window-name
This specifies a different application name and window title for the default command (xterm).

-nols Normally xon passes the -ls option to the remote xterm; this option suspends that behaviour.

-screen screen-no
This changes the screen number of the DISPLAY variable passed to the remote command.

-user user-name
By default, xon simply uses rsh to connect to the remote machine using the same user name as on the local machine. This option cause xon to specify an alternative user name. This will not work unless you have authorization to access the remote account via rsh by placing an appropriate entry in the remote users .rhosts file.

BUGS

Xon can get easily confused when the remote-host, user-name or various environment variable values contain white space.

Xon has no way to send the appropriate X authorization information to the remote host.

NAME

xpr – print an X window dump

SYNOPSIS

xpr [**–device** *devtype*] [**–scale** *scale*] [**–height** *inches*] [**–width** *inches*] [**–left** *inches*] [**–top** *inches*] [**–header** *string*] [**–trailer** *string*] [**–landscape**] [**–portrait**] [**–plane** *number*] [**–gray**] [**–rv**] [**–compact**] [**–output** *filename*] [**–append** *filename*] [**–noff**] [**–split** *n*] [**–psfig**] [**–density** *dpi*] [**–cutoff** *level*] [**–noposition**] [**–gamma** *correction*] [**–render** *algorithm*] [**–slide**] [*filename*]

DESCRIPTION

xpr takes as input a window dump file produced by *xwd(1)* and formats it for output on PostScript printers, the Digital LN03 or LA100, the IBM PP3812 page printer, the HP LaserJet (or other PCL printers), or the HP PaintJet. If no file argument is given, the standard input is used. By default, *xpr* prints the largest possible representation of the window on the output page. Options allow the user to add headers and trailers, specify margins, adjust the scale and orientation, and append multiple window dumps to a single output file. Output is to standard output unless **–output** is specified.

Command Options

–device *devtype*

Specifies the device on which the file will be printed. Currently supported:

la100	Digital LA100
ljet	HP LaserJet series and other monochrome PCL devices such as ThinkJet, QuietJet, RuggedWriter, HP2560 series, and HP2930 series printers
ln03	Digital LN03
pjet	HP PaintJet (color mode)
pjetxl	HP HP PaintJet XL Color Graphics Printer (color mode)
pp	IBM PP3812
ps	PostScript printer

The default is PostScript. **-device lw** (LaserWriter) is equivalent to -device ps and is provided only for backwards compatibility.

–scale *scale*

Affects the size of the window on the page. The PostScript, LN03, and HP printers are able to translate each bit in a window pixel map into a grid of a specified size. For example each bit might translate into a 3x3 grid. This would be specified by **–scale** *3*. By default a window is printed with the largest scale that will fit onto the page for the specified orientation.

–height *inches*

Specifies the maximum height of the page.

–width *inches*

Specifies the maximum width of the page.

–left *inches*

Specifies the left margin in inches. Fractions are allowed. By default the window is centered in the page.

–top *inches*

Specifies the top margin for the picture in inches. Fractions are allowed.

–header *string*

Specifies a header string to be printed above the window.

–trailer *string*

Specifies a trailer string to be printed below the window.

–landscape

Forces the window to printed in landscape mode. By default a window is printed such that its longest side follows the long side of the paper.

–plane *number*
Specifies which bit plane to use in an image. The default is to use the entire image and map values into black and white based on color intensities.

–gray *2 | 3 | 4*
Uses a simple 2x2, 3x3, or 4x4 gray scale conversion on a color image, rather than mapping to strictly black and white. This doubles, triples, or quadruples the effective width and height of the image.

–portrait
Forces the window to be printed in portrait mode. By default a window is printed such that its longest side follows the long side of the paper.

–rv Forces the window to be printed in reverse video.

–compact
Uses simple run-length encoding for compact representation of windows with lots of white pixels.

–output *filename*
Specifies an output file name. If this option is not specified, standard output is used.

–append *filename*
Specifies a filename previously produced by *xpr* to which the window is to be appended.

–noff When specified in conjunction with **–append**, the window will appear on the same page as the previous window.

–split *n* This option allows the user to split a window onto several pages. This might be necessary for very large windows that would otherwise cause the printer to overload and print the page in an obscure manner.

–psfig Suppress translation of the PostScript picture to the center of the page.

–density *dpi*
Indicates what dot-per-inch density should be used by the HP printer.

–cutoff *level*
Changes the intensity level where colors are mapped to either black or white for monochrome output on a LaserJet printer. The *level* is expressed as percentage of full brightness. Fractions are allowed.

–noposition
This option causes header, trailer, and image positioning command generation to be bypassed for LaserJet, PaintJet and PaintJet XL printers.

–gamma *correction*
This changes the intensity of the colors printed by PaintJet XL printer. The *correction* is a floating point value in the range 0.00 to 3.00. Consult the operator's manual to determine the correct value for the specific printer.

–render *algorithm*
This allows PaintJet XL printer to render the image with the best quality versus performance tradeoff. Consult the operator's manual to determine which *algorithm*s are available.

–slide This option allows overhead transparencies to be printed using the PaintJet and PaintJet XL printers.

SEE ALSO

xwd(1), xwud(1), X(1)

LIMITATIONS

The current version of *xpr* can generally print out on the LN03 most X windows that are not larger than two-thirds of the screen. For example, it will be able to print out a large Emacs window, but it will usually fail when trying to print out the entire screen. The LN03 has memory limitations that can cause it to incorrectly print very large or complex windows. The two most common errors encountered are ‘‘band too complex’’ and ‘‘page memory exceeded.’’ In the first case, a window may have a particular six pixel row that contains too many changes (from black to white to black). This will cause the printer to drop part of the line and possibly parts of the rest of the page. The printer will flash the number ‘1’ on its front panel when this problem occurs. A possible solution to this problem is to increase the scale of the picture, or to split the picture onto two or more pages. The second problem, ‘‘page memory exceeded,’’ will occur if the picture contains too much black, or if the picture contains complex half-tones such as the background color of a display. When this problem occurs the printer will automatically split the picture into two or more pages. It may flash the number ‘5’ on its from panel. There is no easy solution to this problem. It will probably be necessary to either cut and paste, or to rework the application to produce a less complex picture.

There are several limitations on the LA100 support: the picture will always be printed in portrait mode, there is no scaling, and the aspect ratio will be slightly off.

Support for PostScript output currently cannot handle the **-append**, **-noff** or **-split** options.

The **-compact** option is *only* supported for PostScript output. It compresses white space but not black space, so it is not useful for reverse-video windows.

For color images, should map directly to PostScript image support.

HP PRINTERS

If no **–density** is specified on the command line 300 dots per inch will be assumed for *ljet* and 90 dots per inch for *pjet*. Allowable *density* values for a LaserJet printer are 300, 150, 100, and 75 dots per inch. Consult the operator’s manual to determine densities supported by other printers.

If no **–scale** is specified the image will be expanded to fit the printable page area.

The default printable page area is 8x10.5 inches. Other paper sizes can be accommodated using the **–height** and **–width** options.

Note that a 1024x768 image fits the default printable area when processed at 100 dpi with scale=1, the same image can also be printed using 300 dpi with scale=3 but will require considerably more data be transferred to the printer.

xpr may be tailored for use with monochrome PCL printers other than the LaserJet. To print on a ThinkJet (HP2225A) *xpr* could be invoked as:

xpr -density 96 -width 6.667 *filename*

or for black-and-white output to a PaintJet:

xpr -density 180 *filename*

The monochrome intensity of a pixel is computed as 0.30∗R + 0.59∗G + 0.11∗B. If a pixel’s computed intensity is less than the **–cutoff** level it will print as white. This maps light-on-dark display images to black-on-white hardcopy. The default cutoff intensity is 50% of full brightness. Example: specifying **–cutoff 87.5** moves the white/black intensity point to 87.5% of full brightness.

A LaserJet printer must be configured with sufficient memory to handle the image. For a full page at 300 dots per inch approximately 2MB of printer memory is required.

Color images are produced on the PaintJet at 90 dots per inch. The PaintJet is limited to sixteen colors from its 330 color palette on each horizontal print line. *xpr* will issue a warning message if more than sixteen colors are encountered on a line. *xpr* will program the PaintJet for the first sixteen colors encountered on each line and use the nearest matching programmed value for other colors present on the line.

Specifying the **–rv**, reverse video, option for the PaintJet will cause black and white to be interchanged on the output image. No other colors are changed.

Multiplane images must be recorded by *xwd* in *ZPixmap* format. Single plane (monochrome) images may be in either *XYPixmap* or *ZPixmap* format.

Some PCL printers do not recognize image positioning commands. Output for these printers will not be centered on the page and header and trailer strings may not appear where expected.

The **–gamma** and **-render** options are supported only on the PaintJet XL printers.

The **–slide** option is not supported for LaserJet printers.

The **–split** option is not supported for HP printers.

The **–gray** option is not supported for HP or IBM printers.

COPYRIGHT

AUTHORS

Michael R. Gretzinger, MIT Project Athena, Jose Capo, MIT Project Athena (PP3812 support), Marvin Solomon, University of Wisconsin, Bob Scheifler, MIT, Angela Bock and E. Mike Durbin, Rich Inc. (grayscale), Larry Rupp, HP (HP printer support).

NAME

xprop - property displayer for X

SYNOPSIS

xprop [-help] [-grammar] [-id *id*] [-root] [-name *name*] [-frame] [-font *font*] [-display *display*] [-len *n*] [-notype] [-fs *file*] [-remove *property-name*] [-spy] [-f *atom format* [*dformat*]]∗ [*format* [*dformat*] *atom*]∗

SUMMARY

The *prop* utility is for displaying window and font properties in an X server. One window or font is selected using the command line arguments or possibly in the case of a window, by clicking on the desired window. A list of properties is then given, possibly with formatting information.

OPTIONS

-help Print out a summary of command line options.

-grammar
Print out a detailed grammar for all command line options.

-id *id* This argument allows the user to select window *id* on the command line rather than using the pointer to select the target window. This is very useful in debugging X applications where the target window is not mapped to the screen or where the use of the pointer might be impossible or interfere with the application.

-name *name*
This argument allows the user to specify that the window named *name* is the target window on the command line rather than using the pointer to select the target window.

-font *font*
This argument allows the user to specify that the properties of font *font* should be displayed.

-root This argument specifies that X's root window is the target window. This is useful in situations where the root window is completely obscured.

-display *display*
This argument allows you to specify the server to connect to; see *X(1)*.

-len *n* Specifies that at most *n* bytes of any property should be read or displayed.

-notype Specifies that the type of each property should not be displayed.

-fs *file* Specifies that file *file* should be used as a source of more formats for properties.

-frame Specifies that when selecting a window by hand (i.e. if none of **-name**, **-root**, or **-id** are given), look at the window manager frame (if any) instead of looking for the client window.

-remove *property-name*
Specifies the name of a property to be removed from the indicated window.

-spy Examine window properties forever, looking for property change events.

-f *name format* **[***dformat***]**
Specifies that the *format* for *name* should be *format* and that the *dformat* for *name* should be *dformat*. If *dformat* is missing, " = $0+\n" is assumed.

DESCRIPTION

For each of these properties, its value on the selected window or font is printed using the supplied formatting information if any. If no formatting information is supplied, internal defaults are used. If a property is not defined on the selected window or font, "not defined" is printed as the value for that property. If no property list is given, all the properties possessed by the selected window or font are printed.

A window may be selected in one of four ways. First, if the desired window is the root window, the -root argument may be used. If the desired window is not the root window, it may be selected in two ways on the command line, either by id number such as might be obtained from *xwininfo*, or by name if the window possesses a name. The -id argument selects a window by id number in either decimal or hex (must start with 0x) while the -name argument selects a window by name.

The last way to select a window does not involve the command line at all. If none of -font, -id, -name, and -root are specified, a crosshairs cursor is displayed and the user is allowed to choose any visible window by pressing any pointer button in the desired window. If it is desired to display properties of a font as opposed to a window, the -font argument must be used.

Other than the above four arguments and the -help argument for obtaining help, and the -grammar argument for listing the full grammar for the command line, all the other command line arguments are used in specifying both the format of the properties to be displayed and how to display them. The -len *n* argument specifies that at most *n* bytes of any given property will be read and displayed. This is useful for example when displaying the cut buffer on the root window which could run to several pages if displayed in full.

Normally each property name is displayed by printing first the property name then its type (if it has one) in parentheses followed by its value. The -notype argument specifies that property types should not be displayed. The -fs argument is used to specify a file containing a list of formats for properties while the -f argument is used to specify the format for one property.

The formatting information for a property actually consists of two parts, a *format* and a *dformat*. The *format* specifies the actual formatting of the property (i.e., is it made up of words, bytes, or longs?, etc.) while the *dformat* specifies how the property should be displayed.

The following paragraphs describe how to construct *format*s and *dformat*s. However, for the vast majority of users and uses, this should not be necessary as the built in defaults contain the *format*s and *dformat*s necessary to display all the standard properties. It should only be necessary to specify *format*s and *dformat*s if a new property is being dealt with or the user dislikes the standard display format. New users especially are encouraged to skip this part.

A *format* consists of one of 0, 8, 16, or 32 followed by a sequence of one or more format characters. The 0, 8, 16, or 32 specifies how many bits per field there are in the property. Zero is a special case meaning use the field size information associated with the property itself. (This is only needed for special cases like type INTEGER which is actually three different types depending on the size of the fields of the property)

A value of 8 means that the property is a sequence of bytes while a value of 16 would mean that the property is a sequence of words. The difference between these two lies in the fact that the sequence of words will be byte swapped while the sequence of bytes will not be when read by a machine of the opposite byte order of the machine that originally wrote the property. For more information on how properties are formatted and stored, consult the Xlib manual.

Once the size of the fields has been specified, it is necessary to specify the type of each field (i.e., is it an integer, a string, an atom, or what?) This is done using one format character per field. If there are more fields in the property than format characters supplied, the last character will be repeated as many times as necessary for the extra fields. The format characters and their meaning are as follows:

a — The field holds an atom number. A field of this type should be of size 32.

b — The field is an boolean. A 0 means false while anything else means true.

c — The field is an unsigned number, a cardinal.

i — The field is a signed integer.

m — The field is a set of bit flags, 1 meaning on.

s — This field and the next ones until either a 0 or the end of the property represent a sequence of bytes. This format character is only usable with a field size of 8 and is most often used to represent a string.

x — The field is a hex number (like ’c’ but displayed in hex - most useful for displaying window ids and the like)

An example *format* is 32ica which is the format for a property of three fields of 32 bits each, the first holding a signed integer, the second an unsigned integer, and the third an atom.

The format of a *dformat* unlike that of a *format* is not so rigid. The only limitations on a *dformat* is that one may not start with a letter or a dash. This is so that it can be distinguished from a property name or an argument. A *dformat* is a text string containing special characters instructing that various fields be printed at various points in a manner similar to the formatting string used by printf. For example, the *dformat* " is ($0, $1 \)\n" would render the POINT 3, -4 which has a *format* of 32ii as " is (3, -4)\n".

Any character other than a $, ?, \, or a (in a *dformat* prints as itself. To print out one of $, ?, \, or (precede it by a \. For example, to print out a $, use \$. Several special backslash sequences are provided as shortcuts. \n will cause a newline to be displayed while \t will cause a tab to be displayed. *o* where *o* is an octal number will display character number *o*.

A $ followed by a number *n* causes field number *n* to be displayed. The format of the displayed field depends on the formatting character used to describe it in the corresponding *format*. I.e., if a cardinal is described by 'c' it will print in decimal while if it is described by a 'x' it is displayed in hex.

If the field is not present in the property (this is possible with some properties), <field not available> is displayed instead. $*n*+ will display field number *n* then a comma then field number *n*+1 then another comma then ... until the last field defined. If field *n* is not defined, nothing is displayed. This is useful for a property that is a list of values.

A ? is used to start a conditional expression, a kind of if-then statement. ?*exp*(*text*) will display *text* if and only if *exp* evaluates to non-zero. This is useful for two things. First, it allows fields to be displayed if and only if a flag is set. And second, it allows a value such as a state number to be displayed as a name rather than as just a number. The syntax of *exp* is as follows:

exp ::= *term* | *term*=*exp* | !*exp*

term ::= *n* | $*n* | m*n*

The ! operator is a logical ''not'', changing 0 to 1 and any non-zero value to 0. = is an equality operator. Note that internally all expressions are evaluated as 32 bit numbers so -1 is not equal to 65535. = returns 1 if the two values are equal and 0 if not. *n* represents the constant value *n* while $*n* represents the value of field number *n*. m*n* is 1 if flag number *n* in the first field having format character 'm' in the corresponding *format* is 1, 0 otherwise.

Examples: ?m3(count: $3\n) displays field 3 with a label of count if and only if flag number 3 (count starts at 0!) is on. ?$2=0(True)?!$2=0(False) displays the inverted value of field 2 as a boolean.

In order to display a property, *xprop* needs both a *format* and a *dformat*. Before *xprop* uses its default values of a *format* of 32x and a *dformat* of " = { $0+ }\n", it searches several places in an attempt to find more specific formats. First, a search is made using the name of the property. If this fails, a search is made using the type of the property. This allows type STRING to be defined with one set of formats while allowing property WM_NAME which is of type STRING to be defined with a different format. In this way, the display formats for a given type can be overridden for specific properties.

The locations searched are in order: the format if any specified with the property name (as in 8x WM_NAME), the formats defined by -f options in last to first order, the contents of the file specified by the -fs option if any, the contents of the file specified by the environmental variable XPROPFORMATS if any, and finally *xprop*'s built in file of formats.

The format of the files referred to by the -fs argument and the XPROPFORMATS variable is one or more lines of the following form:

name format [*dformat*]

Where *name* is either the name of a property or the name of a type, *format* is the *format* to be used with *name* and *dformat* is the *dformat* to be used with *name*. If *dformat* is not present, " = $0+\n" is assumed.

EXAMPLES

To display the name of the root window: *xprop* -root WM_NAME

To display the window manager hints for the clock: *xprop* -name xclock WM_HINTS

To display the start of the cut buffer: *xprop* -root -len 100 CUT_BUFFER0

To display the point size of the fixed font: *xprop* -font fixed POINT_SIZE

To display all the properties of window # 0x200007: *xprop* -id 0x200007

ENVIRONMENT

DISPLAY

To get default display.

XPROPFORMATS

Specifies the name of a file from which additional formats are to be obtained.

SEE ALSO

X(1), xwininfo(1)

COPYRIGHT

AUTHOR

Mark Lillibridge, MIT Project Athena

NAME

xrdb - X server resource database utility

SYNOPSIS

xrdb [-option ...] [*filename*]

DESCRIPTION

Xrdb is used to get or set the contents of the RESOURCE_MANAGER property on the root window of screen 0, or the SCREEN_RESOURCES property on the root window of any or all screens, or everything combined. You would normally run this program from your X startup file.

Most X clients use the RESOURCE_MANAGER and SCREEN_RESOURCES properties to get user preferences about color, fonts, and so on for applications. Having this information in the server (where it is available to all clients) instead of on disk, solves the problem in previous versions of X that required you to maintain *defaults* files on every machine that you might use. It also allows for dynamic changing of defaults without editing files.

The RESOURCE_MANAGER property is used for resources that apply to all screens of the display. The SCREEN_RESOURCES property on each screen specifies additional (or overriding) resources to be used for that screen. (When there is only one screen, SCREEN_RESOURCES is normally not used, all resources are just placed in the RESOURCE_MANAGER property.)

The file specified by *filename* (or the contents from standard input if - or no filename is given) is optionally passed through the C preprocessor with the following symbols defined, based on the capabilities of the server being used:

BITS_PER_RGB=num
the number of significant bits in an RGB color specification. This is the log base 2 of the number of distinct shades of each primary that the hardware can generate. Note that it usually is not related to PLANES.

CLASS=visualclass
one of StaticGray, GrayScale, StaticColor, PseudoColor, TrueColor, DirectColor. This is the visual class of the root window of the default screen.

COLOR defined only if CLASS is one of StaticColor, PseudoColor, TrueColor, or DirectColor.

HEIGHT=num
the height of the default screen in pixels.

SERVERHOST=hostname
the hostname portion of the display to which you are connected.

HOST=hostname
the same as **SERVERHOST**.

CLIENTHOST=hostname
the name of the host on which *xrdb* is running.

PLANES=num
the number of bit planes (the depth) of the root window of the default screen.

RELEASE=num
the vendor release number for the server. The interpretation of this number will vary depending on VENDOR.

REVISION=num
the X protocol minor version supported by this server (currently 0).

VERSION=num
the X protocol major version supported by this server (should always be 11).

VENDOR=vendor
a string specifying the vendor of the server.

WIDTH=num
the width of the default screen in pixels.

X_RESOLUTION=num
the x resolution of the default screen in pixels per meter.

Y_RESOLUTION=num
the y resolution of the default screen in pixels per meter.

Lines that begin with an exclamation mark (!) are ignored and may be used as comments.

Note that since *xrdb* can read from standard input, it can be used to the change the contents of properties directly from a terminal or from a shell script.

OPTIONS

xrdb program accepts the following options:

–help This option (or any unsupported option) will cause a brief description of the allowable options and parameters to be printed.

–display *display*
This option specifies the X server to be used; see *X(1)*. It also specifies the screen to use for the *-screen* option, and it specifies the screen from which preprocessor symbols are derived for the *-global* option.

–all This option indicates that operation should be performed on the screen-independent resource property (RESOURCE_MANAGER), as well as the screen-specific property (SCREEN_RESOURCES) on every screen of the display. For example, when used in conjunction with *-query*, the contents of all properties are output. For *-load* and *-merge*, the input file is processed once for each screen. The resources which occur in common in the output for every screen are collected, and these are applied as the screen-independent resources. The remaining resources are applied for each individual per-screen property. This the default mode of operation.

–global This option indicates that the operation should only be performed on the screen-independent RESOURCE_MANAGER property.

–screen This option indicates that the operation should only be performed on the SCREEN_RESOURCES property of the default screen of the display.

–screens This option indicates that the operation should be performed on the SCREEN_RESOURCES property of each screen of the display. For *-load* and *-merge*, the input file is processed for each screen.

–n This option indicates that changes to the specified properties (when used with *-load* or *-merge*) or to the resource file (when used with *-edit*) should be shown on the standard output, but should not be performed.

–quiet This option indicates that warning about duplicate entries should not be displayed.

-cpp *filename*
This option specifies the pathname of the C preprocessor program to be used. Although *xrdb* was designed to use CPP, any program that acts as a filter and accepts the -D, -I, and -U options may be used.

-nocpp This option indicates that *xrdb* should not run the input file through a preprocessor before loading it into properties.

–symbols
This option indicates that the symbols that are defined for the preprocessor should be printed onto the standard output.

–query This option indicates that the current contents of the specified properties should be printed onto the standard output. Note that since preprocessor commands in the input resource file are part of the input file, not part of the property, they won't appear in the output from this option. The **–edit**

option can be used to merge the contents of properties back into the input resource file without damaging preprocessor commands.

–load This option indicates that the input should be loaded as the new value of the specified properties, replacing whatever was there (i.e. the old contents are removed). This is the default action.

–merge This option indicates that the input should be merged with, instead of replacing, the current contents of the specified properties. Note that this option does a lexicographic sorted merge of the two inputs, which is almost certainly not what you want, but remains for backward compatibility.

–remove This option indicates that the specified properties should be removed from the server.

–retain This option indicates that the server should be instructed not to reset if *xrdb* is the first client. This never be necessary under normal conditions, since *xdm* and *xinit* always act as the first client.

–edit *filename*
This option indicates that the contents of the specified properties should be edited into the given file, replacing any values already listed there. This allows you to put changes that you have made to your defaults back into your resource file, preserving any comments or preprocessor lines.

–backup *string*
This option specifies a suffix to be appended to the filename used with **–edit** to generate a backup file.

–D*name[=value]*
This option is passed through to the preprocessor and is used to define symbols for use with conditionals such as *#ifdef.*

–U*name* This option is passed through to the preprocessor and is used to remove any definitions of this symbol.

–I*directory*
This option is passed through to the preprocessor and is used to specify a directory to search for files that are referenced with *#include.*

FILES

Generalizes *~/.Xdefaults* files.

SEE ALSO

X(1), Xlib Resource Manager documentation, Xt resource documentation

ENVIRONMENT

DISPLAY
to figure out which display to use.

BUGS

The default for no arguments should be to query, not to overwrite, so that it is consistent with other programs.

COPYRIGHT

AUTHORS

Bob Scheifler, Phil Karlton, rewritten from the original by Jim Gettys

NAME

xrefresh - refresh all or part of an X screen

SYNOPSIS

xrefresh [-option ...]

DESCRIPTION

Xrefresh is a simple X program that causes all or part of your screen to be repainted. This is useful when system messages have messed up your screen. *Xrefresh* maps a window on top of the desired area of the screen and then immediately unmaps it, causing refresh events to be sent to all applications. By default, a window with no background is used, causing all applications to repaint ‘‘smoothly.’’ However, the various options can be used to indicate that a solid background (of any color) or the root window background should be used instead.

ARGUMENTS

–white Use a white background. The screen just appears to flash quickly, and then repaint.

–black Use a black background (in effect, turning off all of the electron guns to the tube). This can be somewhat disorienting as everything goes black for a moment.

–solid *color*
Use a solid background of the specified color. Try green.

–root Use the root window background.

–none This is the default. All of the windows simply repaint.

–geometry *WxH+X+Y*
Specifies the portion of the screen to be repainted; see *X(1)*.

–display *display*
This argument allows you to specify the server and screen to refresh; see *X(1)*.

X DEFAULTS

The *xrefresh* program uses the routine *XGetDefault(3X)* to read defaults, so its resource names are all capitalized.

Black, **White**, **Solid**, **None**, **Root**
Determines what sort of window background to use.

Geometry
Determines the area to refresh. Not very useful.

ENVIRONMENT

DISPLAY - To get default host and display number.

SEE ALSO

X(1)

BUGS

It should have just one default type for the background.

COPYRIGHT

AUTHORS

Jim Gettys, Digital Equipment Corp., MIT Project Athena

NAME

xset - user preference utility for X

SYNOPSIS

xset [-display *display*] [-b] [b on/off] [b [*volume* [*pitch* [*duration*]]] [[-]bc] [-c] [c on/off] [c [*volume*]] [[-+]fp[-+=] *path*[,*path*[,...]]] [fp default] [fp rehash] [[-]led [*integer*]] [led on/off] [m[ouse] [*accel_mult*[/*accel_div*] [*threshold*]]] [m[ouse] default] [p *pixel color*] [[-]r] [r on/off] [s [*length* [*period*]]] [s blank/noblank] [s expose/noexpose] [s on/off] [s default] [q]

DESCRIPTION

This program is used to set various user preference options of the display.

OPTIONS

–display *display*
This option specifies the server to use; see *X(1)*.

b The **b** option controls bell volume, pitch and duration. This option accepts up to three numerical parameters, a preceding dash(-), or a ’on/off’ flag. If no parameters are given, or the ’on’ flag is used, the system defaults will be used. If the dash or ’off’ are given, the bell will be turned off. If only one numerical parameter is given, the bell volume will be set to that value, as a percentage of its maximum. Likewise, the second numerical parameter specifies the bell pitch, in hertz, and the third numerical parameter specifies the duration in milliseconds. Note that not all hardware can vary the bell characteristics. The X server will set the characteristics of the bell as closely as it can to the user’s specifications.

bc The **bc** option controls *bug compatibility* mode in the server, if possible; a preceding dash(-) disables the mode, otherwise the mode is enabled. Various pre-R4 clients pass illegal values in some protocol requests, and pre-R4 servers did not correctly generate errors in these cases. Such clients, when run against an R4 server, will terminate abnormally or otherwise fail to operate correctly. Bug compatibility mode explicitly reintroduces certain bugs into the X server, so that many such clients can still be run. This mode should be used with care; new application development should be done with this mode disabled. The server must support the MIT-SUNDRY-NONSTANDARD protocol extension in order for this option to work.

c The **c** option controls key click. This option can take an optional value, a preceding dash(-), or an ’on/off’ flag. If no parameter or the ’on’ flag is given, the system defaults will be used. If the dash or ’off’ flag is used, keyclick will be disabled. If a value from 0 to 100 is given, it is used to indicate volume, as a percentage of the maximum. The X server will set the volume to the nearest value that the hardware can support.

fp= *path*,...
The **fp=** sets the font path to the directories given in the path argument. The directories are interpreted by the server, not by the client, and are server-dependent. Directories that do not contain font databases created by *mkfontdir* will be ignored by the server.

fp default
The **default** argument causes the font path to be reset to the server’s default.

fp rehash
The **rehash** argument causes the server to reread the font databases in the current font path. This is generally only used when adding new fonts to a font directory (after running *mkfontdir* to recreate the font database).

–fp or **fp–**
The **–fp** and **fp–** options remove elements from the current font path. They must be followed by a comma-separated list of directories.

+fp or **fp+**
This **+fp** and **fp+** options prepend and append elements to the current font path, respectively. They must be followed by a comma-separated list of directories.

led The **led** option controls the keyboard LEDs. This controls the turning on or off of one or all of the LEDs. It accepts an optional integer, a preceding dash(-) or an ’on/off’ flag. If no parameter or the ’on’ flag is given, all LEDs are turned on. If a preceding dash or the flag ’off’ is given, all LEDs are turned off. If a value between 1 and 32 is given, that LED will be turned on or off depending on the existence of a preceding dash. A common LED which can be controlled is the ‘‘Caps Lock’’ LED. ‘‘xset led 3’’ would turn led #3 on. ‘‘xset -led 3’’ would turn it off. The particular LED values may refer to different LEDs on different hardware.

m The **m** option controls the mouse parameters. The parameters for the mouse are ‘acceleration’ and ‘threshold’. The acceleration can be specified as an integer, or as a simple fraction. The mouse, or whatever pointer the machine is connected to, will go ‘acceleration’ times as fast when it travels more than ‘threshold’ pixels in a short time. This way, the mouse can be used for precise alignment when it is moved slowly, yet it can be set to travel across the screen in a flick of the wrist when desired. One or both parameters for the **m** option can be omitted, but if only one is given, it will be interpreted as the acceleration. If no parameters or the flag ’default’ is used, the system defaults will be set.

p The **p** option controls pixel color values. The parameters are the color map entry number in decimal, and a color specification. The root background colors may be changed on some servers by altering the entries for BlackPixel and WhitePixel. Although these are often 0 and 1, they need not be. Also, a server may choose to allocate those colors privately, in which case an error will be generated. The map entry must not be a read-only color, or an error will result.

r The **r** option controls the autorepeat. If a preceding dash or the ’off’ flag is used, autorepeat will be disabled. If no parameters or the ’on’ flag is used, autorepeat will be enabled.

s The **s** option lets you set the screen saver parameters. This option accepts up to two numerical parameters, a ’blank/noblank’ flag, an ’expose/noexpose’ flag, an ’on/off’ flag, or the ’default’ flag. If no parameters or the ’default’ flag is used, the system will be set to its default screen saver characteristics. The ’on/off’ flags simply turn the screen saver functions on or off. The ’blank’ flag sets the preference to blank the video (if the hardware can do so) rather than display a background pattern, while ’noblank’ sets the preference to display a pattern rather than blank the video. The ’expose’ flag sets the preference to allow window exposures (the server can freely discard window contents), while ’noexpose’ sets the preference to disable screen saver unless the server can regenerate the screens without causing exposure events. The length and period parameters for the screen saver function determines how long the server must be inactive for screen saving to activate, and the period to change the background pattern to avoid burn in. The arguments are specified in seconds. If only one numerical parameter is given, it will be used for the length.

q The **q** option gives you information on the current settings.

These settings will be reset to default values when you log out.

Note that not all X implementations are guaranteed to honor all of these options.

SEE ALSO

X(1), Xserver(1), xmodmap(1), xrdb(1), xsetroot(1)

COPYRIGHT

AUTHOR

Bob Scheifler, MIT Laboratory for Computer Science
David Krikorian, MIT Project Athena (X11 version)

NAME

xsetroot – root window parameter setting utility for X

SYNOPSIS

xsetroot [-help] [-def] [-display *display*] [-cursor *cursorfile maskfile*] [-cursor_name *cursorname*] [-bitmap *filename*] [-mod *x y*] [-gray] [-grey] [-fg *color*] [-bg *color*] [-rv] [-solid *color*] [-name *string*]

DESCRIPTION

The *setroot* program allows you to tailor the appearance of the background ("root") window on a workstation display running X. Normally, you experiment with *xsetroot* until you find a personalized look that you like, then put the *xsetroot* command that produces it into your X startup file. If no options are specified, or if *-def* is specified, the window is reset to its default state. The *-def* option can be specified along with other options and only the non-specified characteristics will be reset to the default state.

Only one of the background color/tiling changing options (-solid, -gray, -grey, -bitmap, and -mod) may be specified at a time.

OPTIONS

The various options are as follows:

-help Print a usage message and exit.

-def Reset unspecified attributes to the default values. (Restores the background to the familiar gray mesh and the cursor to the hollow x shape.)

-cursor *cursorfile maskfile*
This lets you change the pointer cursor to whatever you want when the pointer cursor is outside of any window. Cursor and mask files are bitmaps (little pictures), and can be made with the *bitmap(1)* program. You probably want the mask file to be all black until you get used to the way masks work.

-cursor_name *cursorname*
This lets you change the pointer cursor to one of the standard cursors from the cursor font. Refer to appendix B of the X protocol for the names (except that the XC_ prefix is elided for this option).

-bitmap *filename*
Use the bitmap specified in the file to set the window pattern. You can make your own bitmap files (little pictures) using the *bitmap(1)* program. The entire background will be made up of repeated "tiles" of the bitmap.

-mod *x y*
This is used if you want a plaid-like grid pattern on your screen. x and y are integers ranging from 1 to 16. Try the different combinations. Zero and negative numbers are taken as 1.

-gray Make the entire background gray. (Easier on the eyes.)

-grey Make the entire background grey.

-fg *color*
Use ‘‘color’’ as the foreground color. Foreground and background colors are meaningful only in combination with -cursor, -bitmap, or -mod.

-bg *color*
Use ‘‘color’’ as the background color.

-rv This exchanges the foreground and background colors. Normally the foreground color is black and the background color is white.

-solid *color*
This sets the background of the root window to the specified color. This option is only useful on color servers.

-name *string*

Set the name of the root window to ‘‘string’’. There is no default value. Usually a name is assigned to a window so that the window manager can use a text representation when the window is iconified. This option is unused since you can’t iconify the background.

-display *display*
Specifies the server to connect to; see *X(1)*.

SEE ALSO

X(1), xset(1), xrdb(1)

COPYRIGHT

AUTHOR

Mark Lillibridge, MIT Project Athena

NAME

xstdcmap - X standard colormap utility

SYNOPSIS

xstdcmap [-all] [-best] [-blue] [-default] [-delete *map*] [-display *display*] [-gray] [-green] [-help] [-red] [-verbose]

DESCRIPTION

The *xstdcmap* utility can be used to selectively define standard colormap properties. It is intended to be run from a user's X startup script to create standard colormap definitions in order to facilitate sharing of scarce colormap resources among clients. Where at all possible, colormaps are created with read-only allocations.

OPTIONS

The following options may be used with *xstdcmap*:

–all This option indicates that all six standard colormap properties should be defined on each screen of the display. Not all screens will support visuals under which all six standard colormap properties are meaningful. *xstdcmap* will determine the best allocations and visuals for the colormap properties of a screen. Any previously existing standard colormap properties will be replaced.

–best This option indicates that the RGB_BEST_MAP should be defined.

–blue This option indicates that the RGB_BLUE_MAP should be defined.

–default This option indicates that the RGB_DEFAULT_MAP should be defined.

–delete *map*
This option specifies that a standard colormap property should be removed. *mapP may be one of: default, best, red, green, blue, or gray.*

–display *display*
This option specifies the host and display to use; see *X(1)*.

–gray This option indicates that the RGB_GRAY_MAP should be defined.

–green This option indicates that the RGB_GREEN_MAP should be defined.

–help This option indicates that a brief description of the command line arguments should be printed on the standard error. This will be done whenever an unhandled argument is given to *xstdcmap.*

–red This option indicates that the RGB_RED_MAP should be defined.

–verbose
This option indicates that *xstdcmap* should print logging information as it parses its input and defines the standard colormap properties.

ENVIRONMENT

DISPLAY
to get default host and display number.

SEE ALSO

X(1)

COPYRIGHT

AUTHOR

Donna Converse, MIT X Consortium

NAME

xterm – terminal emulator for X

SYNOPSIS

xterm [–*toolkitoption* ...] [–option ...]

DESCRIPTION

The *xterm* program is a terminal emulator for the X Window System. It provides DEC VT102 and Tektronix 4014 compatible terminals for programs that can't use the window system directly. If the underlying operating system supports terminal resizing capabilities (for example, the SIGWINCH signal in systems derived from 4.3bsd), *xterm* will use the facilities to notify programs running in the window whenever it is resized.

The VT102 and Tektronix 4014 terminals each have their own window so that you can edit text in one and look at graphics in the other at the same time. To maintain the correct aspect ratio (height/width), Tektronix graphics will be restricted to the largest box with a 4014's aspect ratio that will fit in the window. This box is located in the upper left area of the window.

Although both windows may be displayed at the same time, one of them is considered the ''active'' window for receiving keyboard input and terminal output. This is the window that contains the text cursor. The active window can be chosen through escape sequences, the ''VT Options'' menu in the VT102 window, and the ''Tek Options'' menu in the 4014 window.

EMULATIONS

The VT102 emulation is fairly complete, but does not support the blinking character attribute nor the double-wide and double-size character sets. *Termcap*(5) entries that work with *xterm* include ''xterm,'' ''vt102,'' ''vt100'' and ''ansi,'' and *xterm* automatically searches the termcap file in this order for these entries and then sets the ''TERM'' and the ''TERMCAP'' environment variables.

Many of the special *xterm* features (like logging) may be modified under program control through a set of escape sequences different from the standard VT102 escape sequences. (See the *Xterm Control Sequences* document.)

The Tektronix 4014 emulation is also fairly good. Four different font sizes and five different lines types are supported. The Tektronix text and graphics commands are recorded internally by *xterm* and may be written to a file by sending the COPY escape sequence (or through the **Tektronix** menu; see below). The name of the file will be ''**COPY***yy***–***MM***–***dd***.***hh***:***mm***:***ss*'', where *yy*, *MM*, *dd*, *hh*, *mm* and *ss* are the year, month, day, hour, minute and second when the COPY was performed (the file is created in the directory *xterm* is started in, or the home directory for a login *xterm*).

OTHER FEATURES

Xterm automatically highlights the text cursor when the pointer enters the window (selected) and unhighlights it when the pointer leaves the window (unselected). If the window is the focus window, then the text cursor is highlighted no matter where the pointer is.

In VT102 mode, there are escape sequences to activate and deactivate an alternate screen buffer, which is the same size as the display area of the window. When activated, the current screen is saved and replaced with the alternate screen. Saving of lines scrolled off the top of the window is disabled until the normal screen is restored. The *termcap*(5) entry for *xterm* allows the visual editor *vi*(1) to switch to the alternate screen for editing and to restore the screen on exit.

In either VT102 or Tektronix mode, there are escape sequences to change the name of the windows and to specify a new log file name. See *Xterm Control Sequences* for details. Enabling the escape sequence to change the log file name is a compile-time option; by default this escape sequence is ignored for security reasons.

OPTIONS

The *xterm* terminal emulator accepts all of the standard X Toolkit command line options as well as the following (if the option begins with a '**+**' instead of a '**–**', the option is restored to its default value):

–help This causes *xterm* to print out a verbose message describing its options.

–132 Normally, the VT102 DECCOLM escape sequence that switches between 80 and 132 column mode is ignored. This option causes the DECCOLM escape sequence to be recognized, and the *xterm* window will resize appropriately.

–ah This option indicates that *xterm* should always highlight the text cursor. By default, *xterm* will display a hollow text cursor whenever the focus is lost or the pointer leaves the window.

+ah This option indicates that *xterm* should do text cursor highlighting based on focus.

–b *number*
This option specifies the size of the inner border (the distance between the outer edge of the characters and the window border) in pixels. The default is 2.

–cb Set the *vt100* resource **cutToBeginningOfLine** to FALSE.

+cb Set the *vt100* resource **cutToBeginningOfLine** to TRUE.

–cc *characterclassrange***:***value***[,...]**
This sets classes indicated by the given ranges for using in selecting by words. See the section specifying character classes.

–cn This option indicates that newlines should not be cut in line-mode selections.

+cn This option indicates that newlines should be cut in line-mode selections.

–cr *color*
This option specifies the color to use for text cursor. The default is to use the same foreground color that is used for text.

–cu This option indicates that *xterm* should work around a bug in the *curses*(3x) cursor motion package that causes the *more*(1) program to display lines that are exactly the width of the window and are followed by a line beginning with a tab to be displayed incorrectly (the leading tabs are not displayed).

+cu This option indicates that that *xterm* should not work around the *curses*(3x) bug mentioned above.

–e *program [arguments ...]*
This option specifies the program (and its command line arguments) to be run in the *xterm* window. It also sets the window title and icon name to be the basename of the program being executed if neither *–T* nor *–n* are given on the command line. **This must be the last option on the command line.**

–fb *font* This option specifies a font to be used when displaying bold text. This font must be the same height and width as the normal font. If only one of the normal or bold fonts is specified, it will be used as the normal font and the bold font will be produced by overstriking this font. The default is to do overstriking of the normal font.

–im Turn on the **useInsertMode** resource.

+im Turn off the **useInsertMode** resource.

–j This option indicates that *xterm* should do jump scrolling. Normally, text is scrolled one line at a time; this option allows *xterm* to move multiple lines at a time so that it doesn't fall as far behind. Its use is strongly recommended since it make *xterm* much faster when scanning through large amounts of text. The VT100 escape sequences for enabling and disabling smooth scroll as well as the ''VT Options'' menu can be used to turn this feature on or off.

+j This option indicates that *xterm* should not do jump scrolling.

–l This option indicates that *xterm* should send all terminal output to a log file as well as to the screen. This option can be enabled or disabled using the ''VT Options'' menu.

+l This option indicates that *xterm* should not do logging.

–lf *filename*
This option specifies the name of the file to which the output log described above is written. If *file* begins with a pipe symbol (|), the rest of the string is assumed to be a command to be used as the endpoint of a pipe. The ability to log to a pipe is a compile-time option which is disabled by default for security reasons. The default filename is ''**XtermLog.***XXXXX*'' (where *XXXXX* is the process id of *xterm*) and is created in the directory from which *xterm* was started (or the user's home directory in the case of a login window).

–ls This option indicates that the shell that is started in the *xterm* window be a login shell (i.e. the first character of argv[0] will be a dash, indicating to the shell that it should read the user's .login or .profile).

+ls This option indicates that the shell that is started should not be a login shell (i.e. it will be a normal ''subshell'').

–mb This option indicates that *xterm* should ring a margin bell when the user types near the right end of a line. This option can be turned on and off from the ''VT Options'' menu.

+mb This option indicates that margin bell should not be rung.

–mc milliseconds
This option specifies the maximum time between multi-click selections.

–ms *color*
This option specifies the color to be used for the pointer cursor. The default is to use the foreground color.

–nb *number*
This option specifies the number of characters from the right end of a line at which the margin bell, if enabled, will ring. The default is 10.

–rw This option indicates that reverse-wraparound should be allowed. This allows the cursor to back up from the leftmost column of one line to the rightmost column of the previous line. This is very useful for editing long shell command lines and is encouraged. This option can be turned on and off from the ''VT Options'' menu.

+rw This option indicates that reverse-wraparound should not be allowed.

–aw This option indicates that auto-wraparound should be allowed. This allows the cursor to automatically wrap to the beginning of the next line when when it is at the rightmost position of a line and text is output.

+aw This option indicates that auto-wraparound should not be allowed.

–s This option indicates that *xterm* may scroll asynchronously, meaning that the screen does not have to be kept completely up to date while scrolling. This allows *xterm* to run faster when network latencies are very high and is typically useful when running across a very large internet or many gateways.

+s This option indicates that *xterm* should scroll synchronously.

–sb This option indicates that some number of lines that are scrolled off the top of the window should be saved and that a scrollbar should be displayed so that those lines can be viewed. This option may be turned on and off from the ''VT Options'' menu.

+sb This option indicates that a scrollbar should not be displayed.

–sf This option indicates that Sun Function Key escape codes should be generated for function keys.

+sf This option indicates that the standard escape codes should be generated for function keys.

–si This option indicates that output to a window should not automatically reposition the screen to the bottom of the scrolling region. This option can be turned on and off from the ''VT Options'' menu.

+si This option indicates that output to a window should cause it to scroll to the bottom.

–sk This option indicates that pressing a key while using the scrollbar to review previous lines of text should cause the window to be repositioned automatically in the normal position at the bottom of the scroll region.

+sk This option indicates that pressing a key while using the scrollbar should not cause the window to be repositioned.

–sl *number*
This option specifies the number of lines to save that have been scrolled off the top of the screen. The default is 64.

–t This option indicates that *xterm* should start in Tektronix mode, rather than in VT102 mode. Switching between the two windows is done using the ‘‘Options’’ menus.

+t This option indicates that *xterm* should start in VT102 mode.

–tm *string*
This option specifies a series of terminal setting keywords followed by the characters that should be bound to those functions, similar to the *stty* program. Allowable keywords include: intr, quit, erase, kill, eof, eol, swtch, start, stop, brk, susp, dsusp, rprnt, flush, weras, and lnext. Control characters may be specified as ˆchar (e.g. ˆc or ˆu) and ˆ? may be used to indicate delete.

–tn *name*
This option specifies the name of the terminal type to be set in the TERM environment variable. This terminal type must exist in the *termcap(5)* database and should have *li#* and *co#* entries.

–ut This option indicates that *xterm* shouldn’t write a record into the the system log file */etc/utmp*.

+ut This option indicates that *xterm* should write a record into the system log file */etc/utmp*.

–vb This option indicates that a visual bell is preferred over an audible one. Instead of ringing the terminal bell whenever a Control-G is received, the window will be flashed.

+vb This option indicates that a visual bell should not be used.

–wf This option indicates that *xterm* should wait for the window to be mapped the first time before starting the subprocess so that the initial terminal size settings and environment variables are correct. It is the application’s responsibility to catch subsequent terminal size changes.

+wf This option indicates that *xterm* show not wait before starting the subprocess.

–C This option indicates that this window should receive console output. This is not supported on all systems. To obtain console output, you must be the owner of the console device, and you must have read and write permission for it. If you are running X under *xdm* on the console screen you may need to have the session startup and reset programs explicitly change the ownership of the console device in order to get this option to work.

–S*ccn* This option specifies the last two letters of the name of a pseudoterminal to use in slave mode, plus the number of the inherited file descriptor. The option is parsed ‘‘%c%c%d’’. This allows *xterm* to be used as an input and output channel for an existing program and is sometimes used in specialized applications.

The following command line arguments are provided for compatibility with older versions. They may not be supported in the next release as the X Toolkit provides standard options that accomplish the same task.

%*geom* This option specifies the preferred size and position of the Tektronix window. It is shorthand for specifying the ‘‘**tekGeometry*’’ resource.

#*geom* This option specifies the preferred position of the icon window. It is shorthand for specifying the ‘‘**iconGeometry*’’ resource.

–T *string*
This option specifies the title for *xterm*’s windows. It is equivalent to **–title**.

–n *string* This option specifies the icon name for *xterm*'s windows. It is shorthand for specifying the ''**iconName*'' resource. Note that this is not the same as the toolkit option **–name** (see below). The default icon name is the application name.

–r This option indicates that reverse video should be simulated by swapping the foreground and background colors. It is equivalent to **–rv**.

–w *number*
This option specifies the width in pixels of the border surrounding the window. It is equivalent to **–borderwidth** or **–bw**.

The following standard X Toolkit command line arguments are commonly used with *xterm*:

–bg *color*
This option specifies the color to use for the background of the window. The default is ''white.''

–bd *color*
This option specifies the color to use for the border of the window. The default is ''black.''

–bw *number*
This option specifies the width in pixels of the border surrounding the window.

–fg *color* This option specifies the color to use for displaying text. The default is ''black.''

–fn *font* This option specifies the font to be used for displaying normal text. The default is *fixed*.

–name *name*
This option specifies the application name under which resources are to be obtained, rather than the default executable file name. *Name* should not contain ''.'' or ''*'' characters.

–title *string*
This option specifies the window title string, which may be displayed by window managers if the user so chooses. The default title is the command line specified after the **–e** option, if any, otherwise the application name.

–rv This option indicates that reverse video should be simulated by swapping the foreground and background colors.

–geometry *geometry*
This option specifies the preferred size and position of the VT102 window; see *X(1)*.

–display *display*
This option specifies the X server to contact; see *X(1)*.

–xrm *resourcestring*
This option specifies a resource string to be used. This is especially useful for setting resources that do not have separate command line options.

–iconic This option indicates that *xterm* should ask the window manager to start it as an icon rather than as the normal window.

RESOURCES

The program understands all of the core X Toolkit resource names and classes as well as:

iconGeometry (class **IconGeometry)**
Specifies the preferred size and position of the application when iconified. It is not necessarily obeyed by all window managers.

iconName (class **IconName)**
Specifies the icon name. The default is the application name.

termName (class **TermName)**
Specifies the terminal type name to be set in the TERM environment variable.

title (class **Title)**

Specifies a string that may be used by the window manager when displaying this application.

ttyModes (class **TtyModes)**
Specifies a string containing terminal setting keywords and the characters to which they may be bound. Allowable keywords include: intr, quit, erase, kill, eof, eol, swtch, start, stop, brk, susp, dsusp, rprnt, flush, weras, and lnext. Control characters may be specified as ˆchar (e.g. ˆc or ˆu) and ˆ? may be used to indicate Delete. This is very useful for overriding the default terminal settings without having to do an *stty* every time an *xterm* is started.

useInsertMode (class **UseInsertMode)**
Force use of insert mode by adding appropriate entries to the TERMCAP environment variable. This is useful if the system termcap is broken. The default is ‘‘false.’’

utmpInhibit (class **UtmpInhibit)**
Specifies whether or not *xterm* should try to record the user’s terminal in */etc/utmp*.

sunFunctionKeys (class **SunFunctionKeys)**
Specifies whether or not Sun Function Key escape codes should be generated for function keys instead of standard escape sequences.

waitForMap (class **WaitForMap)**
Specifies whether or not *xterm* should wait for the initial window map before starting the subprocess. The default is ‘‘false.’’

The following resources are specified as part of the *vt100* widget (class *VT100*):

allowSendEvents (class **AllowSendEvents)**
Specifies whether or not synthetic key and button events (generated using the X protocol SendEvent request) should be interpreted or discarded. The default is ‘‘false’’ meaning they are discarded. Note that allowing such events creates a very large security hole.

alwaysHighlight (class **AlwaysHighlight)**
Specifies whether or not *xterm* should always display a highlighted text cursor. By default, a hollow text cursor is displayed whenever the pointer moves out of the window or the window loses the input focus.

appcursorDefault (class **AppcursorDefault)**
If ‘‘true,’’ the cursor keys are initially in application mode. The default is ‘‘false.’’

appkeypadDefault (class **AppkeypadDefault)**
If ‘‘true,’’ the keypad keys are initially in application mode. The default is ‘‘false.’’

autoWrap (class **AutoWrap)**
Specifies whether or not auto-wraparound should be enabled. The default is ‘‘true.’’

bellSuppressTime (class **BellSuppressTime)**
Number of milliseconds after a bell command is sent during which additional bells will be suppressed. Default is 200. If set non-zero, additional bells will also be suppressed until the server reports that processing of the first bell has been completed; this feature is most useful with the visible bell.

boldFont (class **BoldFont)**
Specifies the name of the bold font to use instead of overstriking.

c132 (class **C132)**
Specifies whether or not the VT102 DECCOLM escape sequence should be honored. The default is ‘‘false.’’

cutNewline (class **CutNewline)**
If false, triple clicking to select a line does not include the Newline at the end of the line. If true,

the Newline is selected. The default is ‘‘true.’’

cutToBeginningOfLine (class **CutToBeginningOfLine)**
If false, triple clicking to select a line selects only from the current word forward. If true, the entire line is selected. The default is ‘‘true.’’

charClass (class **CharClass)**
Specifies comma-separated lists of character class bindings of the form [*low*-]*high*:*value*. These are used in determining which sets of characters should be treated the same when doing cut and paste. See the section on specifying character classes.

curses (class **Curses)**
Specifies whether or not the last column bug in *curses*(3x) should be worked around. The default is ‘‘false.’’

background (class **Background)**
Specifies the color to use for the background of the window. The default is ‘‘white.’’

foreground (class **Foreground)**
Specifies the color to use for displaying text in the window. Setting the class name instead of the instance name is an easy way to have everything that would normally appear in the text color change color. The default is ‘‘black.’’

cursorColor (class **Foreground)**
Specifies the color to use for the text cursor. The default is ‘‘black.’’

eightBitInput (class **EightBitInput**)
If true, Meta characters input from the keyboard are presented as a single character with the eighth bit turned on. If false, Meta characters are converted into a two-character sequence with the character itself preceded by ESC. The default is ‘‘true.’’

eightBitOutput (class **EightBitOutput**)
Specifies whether or not eight-bit characters sent from the host should be accepted as is or stripped when printed. The default is ‘‘true.’’

font (class **Font)**
Specifies the name of the normal font. The default is ‘‘fixed.’’

font1 (class **Font1)**
Specifies the name of the first alternative font.

font2 (class **Font2)**
Specifies the name of the second alternative font.

font3 (class **Font3)**
Specifies the name of the third alternative font.

font4 (class **Font4)**
Specifies the name of the fourth alternative font.

font5 (class **Font5)**
Specifies the name of the fifth alternative font.

font6 (class **Font6)**
Specifies the name of the sixth alternative font.

geometry (class **Geometry)**
Specifies the preferred size and position of the VT102 window.

internalBorder (class **BorderWidth)**
Specifies the number of pixels between the characters and the window border. The default is 2.

jumpScroll (class **JumpScroll)**
Specifies whether or not jump scroll should be used. The default is ‘‘true.’’

logFile (class **Logfile)**
Specifies the name of the file to which a terminal session is logged. The default is ‘‘**XtermLog.***XXXXX*’’ (where *XXXXX* is the process id of *xterm*).

logging (class **Logging)**
Specifies whether or not a terminal session should be logged. The default is ‘‘false.’’

logInhibit (class **LogInhibit)**
Specifies whether or not terminal session logging should be inhibited. The default is ‘‘false.’’

loginShell (class **LoginShell)**
Specifies whether or not the shell to be run in the window should be started as a login shell. The default is ‘‘false.’’

marginBell (class **MarginBell)**
Specifies whether or not the bell should be run when the user types near the right margin. The default is ‘‘false.’’

multiClickTime (class **MultiClickTime)**
Specifies the maximum time in milliseconds between multi-click select events. The default is 250 milliseconds.

multiScroll (class **MultiScroll)**
Specifies whether or not scrolling should be done asynchronously. The default is ‘‘false.’’

nMarginBell (class **Column)**
Specifies the number of characters from the right margin at which the margin bell should be rung, when enabled.

pointerColor (class **Foreground)**
Specifies the foreground color of the pointer. The default is ‘‘XtDefaultForeground.’’

pointerColorBackground (class **Background)**
Specifies the background color of the pointer. The default is ‘‘XtDefaultBackground.’’

pointerShape (class **Cursor)**
Specifies the name of the shape of the pointer. The default is ‘‘xterm.’’

resizeGravity (class **ResizeGravity)**
Affects the behavior when the window is resized to be taller or shorter. **NorthWest** specifies that the top line of text on the screen stay fixed. If the window is made shorter, lines are dropped from the bottom; if the window is made taller, blank lines are added at the bottom. This is compatible with the behavior in R4. **SouthWest** (the default) specifies that the bottom line of text on the screen stay fixed. If the window is made taller, additional saved lines will be scrolled down onto the screen; if the window is made shorter, lines will be scrolled off the top of the screen, and the top saved lines will be dropped.

reverseVideo (class **ReverseVideo)**
Specifies whether or not reverse video should be simulated. The default is ‘‘false.’’

reverseWrap (class **ReverseWrap)**
Specifies whether or not reverse-wraparound should be enabled. The default is ‘‘false.’’

saveLines (class **SaveLines)**
Specifies the number of lines to save beyond the top of the screen when a scrollbar is turned on. The default is 64.

scrollBar (class **ScrollBar)**
Specifies whether or not the scrollbar should be displayed. The default is ‘‘false.’’

scrollTtyOutput (class **ScrollCond)**
Specifies whether or not output to the terminal should automatically cause the scrollbar to go to the bottom of the scrolling region. The default is ‘‘true.’’

scrollKey (class **ScrollCond)**
Specifies whether or not pressing a key should automatically cause the scrollbar to go to the bottom of the scrolling region. The default is ‘‘false.’’

scrollLines (class **ScrollLines)**
Specifies the number of lines that the *scroll-back* and *scroll-forw* actions should use as a default. The default value is 1.

signalInhibit (class **SignalInhibit)**
Specifies whether or not the entries in the ‘‘Main Options’’ menu for sending signals to *xterm* should be disallowed. The default is ‘‘false.’’

tekGeometry (class **Geometry)**
Specifies the preferred size and position of the Tektronix window.

tekInhibit (class **TekInhibit)**
Specifies whether or not Tektronix mode should be disallowed. The default is ‘‘false.’’

tekSmall (class **TekSmall)**
Specifies whether or not the Tektronix mode window should start in its smallest size if no explicit geometry is given. This is useful when running *xterm* on displays with small screens. The default is ‘‘false.’’

tekStartup (class **TekStartup)**
Specifies whether or not *xterm* should start up in Tektronix mode. The default is ‘‘false.’’

titeInhibit (class **TiteInhibit)**
Specifies whether or not *xterm* should remove remove *ti* and *te* termcap entries (used to switch between alternate screens on startup of many screen-oriented programs) from the TERMCAP string. If set, *xterm* also ignores the escape sequence to switch to the alternate screen.

translations (class **Translations)**
Specifies the key and button bindings for menus, selections, ‘‘programmed strings,’’ etc. See **ACTIONS** below.

visualBell (class **VisualBell)**
Specifies whether or not a visible bell (i.e. flashing) should be used instead of an audible bell when Control-G is received. The default is ‘‘false.’’

The following resources are specified as part of the *tek4014* widget (class *Tek4014*):

width (class **Width)**
Specifies the width of the Tektronix window in pixels.

height (class **Height)**
Specifies the height of the Tektronix window in pixels.

fontLarge (class **Font)**
Specifies the large font to use in the Tektronix window.

font2 (class **Font)**
Specifies font number 2 to use in the Tektronix window.

font3 (class **Font)**
Specifies font number 3 to use in the Tektronix window.

fontSmall (class **Font)**
Specifies the small font to use in the Tektronix window.

initialFont (class **InitialFont)**
Specifies which of the four Tektronix fonts to use initially. Values are the same as for the *set-tek-text* action. The default is ‘‘large.’’

ginTerminator (class **GinTerminator)**
Specifies what character(s) should follow a GIN report or status report. The possibilities are ‘‘none,’’ which sends no terminating characters, ‘‘CRonly,’’ which sends CR, and ‘‘CR&EOT,’’ which sends both CR and EOT. The default is ‘‘none.’’

The resources that may be specified for the various menus are described in the documentation for the Athena **SimpleMenu** widget. The name and classes of the entries in each of the menus are listed below.

The *mainMenu* has the following entries:

securekbd (class **SmeBSB)**
This entry invokes the **secure()** action.

allowsends (class **SmeBSB)**
This entry invokes the **allow-send-events(toggle)** action.

logging (class **SmeBSB)**
This entry invokes the **set-logging(toggle)** action.

redraw (class **SmeBSB)**
This entry invokes the **redraw()** action.

line1 (class **SmeLine)**
This is a separator.

suspend (class **SmeBSB)**
This entry invokes the **send-signal(tstp)** action on systems that support job control.

continue (class **SmeBSB)**
This entry invokes the **send-signal(cont)** action on systems that support job control.

interrupt (class **SmeBSB)**
This entry invokes the **send-signal(int)** action.

hangup (class **SmeBSB)**
This entry invokes the **send-signal(hup)** action.

terminate (class **SmeBSB)**
This entry invokes the **send-signal(term)** action.

kill (class **SmeBSB)**
This entry invokes the **send-signal(kill)** action.

line2 (class **SmeLine)**
This is a separator.

quit (class **SmeBSB)**
This entry invokes the **quit()** action.

The *vtMenu* has the following entries:

scrollbar (class **SmeBSB)**
This entry invokes the **set-scrollbar(toggle)** action.

jumpscroll (class **SmeBSB)**
This entry invokes the **set-jumpscroll(toggle)** action.

reversevideo (class **SmeBSB)**
This entry invokes the **set-reverse-video(toggle)** action.

autowrap (class **SmeBSB)**
This entry invokes the **set-autowrap(toggle)** action.

reversewrap (class **SmeBSB)**

This entry invokes the **set-reversewrap(toggle)** action.

autolinefeed (class **SmeBSB)**
This entry invokes the **set-autolinefeed(toggle)** action.

appcursor (class **SmeBSB)**
This entry invokes the **set-appcursor(toggle)** action.

appkeypad (class **SmeBSB)**
This entry invokes the **set-appkeypad(toggle)** action.

scrollkey (class **SmeBSB)**
This entry invokes the **set-scroll-on-key(toggle)** action.

scrollttyoutput (class **SmeBSB)**
This entry invokes the **set-scroll-on-tty-output(toggle)** action.

allow132 (class **SmeBSB)**
This entry invokes the **set-allow132(toggle)** action.

cursesemul (class **SmeBSB)**
This entry invokes the **set-cursesemul(toggle)** action.

visualbell (class **SmeBSB)**
This entry invokes the **set-visualbell(toggle)** action.

marginbell (class **SmeBSB)**
This entry invokes the **set-marginbell(toggle)** action.

altscreen (class **SmeBSB)**
This entry is currently disabled.

line1 (class **SmeLine)**
This is a separator.

softreset (class **SmeBSB)**
This entry invokes the **soft-reset()** action.

hardreset (class **SmeBSB)**
This entry invokes the **hard-reset()** action.

clearsavedlines (class **SmeBSB)"**
This entry invokes the **clear-saved-lines()** action.

line2 (class **SmeLine)**
This is a separator.

tekshow (class **SmeBSB)**
This entry invokes the **set-visibility(tek,toggle)** action.

tekmode (class **SmeBSB)**
This entry invokes the **set-terminal-type(tek)** action.

vthide (class **SmeBSB)**
This entry invokes the **set-visibility(vt,off)** action.

The *fontMenu* has the following entries:

fontdefault (class **SmeBSB)**
This entry invokes the **set-vt-font(d)** action.

font1 (class **SmeBSB)**
This entry invokes the **set-vt-font(1)** action.

font2 (class **SmeBSB)**
This entry invokes the **set-vt-font(2)** action.

font3 (class **SmeBSB)**
This entry invokes the **set-vt-font(3)** action.

font4 (class **SmeBSB)**
This entry invokes the **set-vt-font(4)** action.

font5 (class **SmeBSB)**
This entry invokes the **set-vt-font(5)** action.

font6 (class **SmeBSB)**
This entry invokes the **set-vt-font(6)** action.

fontescape (class **SmeBSB)**
This entry invokes the **set-vt-font(e)** action.

fontsel (class **SmeBSB)**
This entry invokes the **set-vt-font(s)** action.

The *tekMenu* has the following entries:

tektextlarge (class **SmeBSB)**
This entry invokes the **set-tek-text(l)** action.

tektext2 (class **SmeBSB)**
This entry invokes the **set-tek-text(2)** action.

tektext3 (class **SmeBSB)**
This entry invokes the **set-tek-text(3)** action.

tektextsmall (class **SmeBSB)**
This entry invokes the **set-tek-text(s)** action.

line1 (class **SmeLine)**
This is a separator.

tekpage (class **SmeBSB)**
This entry invokes the **tek-page()** action.

tekreset (class **SmeBSB)**
This entry invokes the **tek-reset()** action.

tekcopy (class **SmeBSB)**
This entry invokes the **tek-copy()** action.

line2 (class **SmeLine)**
This is a separator.

vtshow (class **SmeBSB)**
This entry invokes the **set-visibility(vt,toggle)** action.

vtmode (class **SmeBSB)**
This entry invokes the **set-terminal-type(vt)** action.

tekhide (class **SmeBSB)**
This entry invokes the **set-visibility(tek,toggle)** action.

The following resources are useful when specified for the Athena Scrollbar widget:

thickness (class **Thickness)**
Specifies the width in pixels of the scrollbar.

background (class **Background)**
Specifies the color to use for the background of the scrollbar.

foreground (class **Foreground)**

Specifies the color to use for the foreground of the scrollbar. The ''thumb'' of the scrollbar is a simple checkerboard pattern alternating pixels for foreground and background color.

POINTER USAGE

Once the VT102 window is created, *xterm* allows you to select text and copy it within the same or other windows.

The selection functions are invoked when the pointer buttons are used with no modifiers, and when they are used with the ''shift'' key. The assignment of the functions described below to keys and buttons may be changed through the resource database; see **ACTIONS** below.

Pointer button one (usually left) is used to save text into the cut buffer. Move the cursor to beginning of the text, and then hold the button down while moving the cursor to the end of the region and releasing the button. The selected text is highlighted and is saved in the global cut buffer and made the PRIMARY selection when the button is released. Double-clicking selects by words. Triple-clicking selects by lines. Quadruple-clicking goes back to characters, etc. Multiple-click is determined by the time from button up to button down, so you can change the selection unit in the middle of a selection. If the key/button bindings specify that an X selection is to be made, *xterm* will leave the selected text highlighted for as long as it is the selection owner.

Pointer button two (usually middle) 'types' (pastes) the text from the PRIMARY selection, if any, otherwise from the cut buffer, inserting it as keyboard input.

Pointer button three (usually right) extends the current selection. (Without loss of generality, you can swap ''right'' and ''left'' everywhere in the rest of this paragraph.) If pressed while closer to the right edge of the selection than the left, it extends/contracts the right edge of the selection. If you contract the selection past the left edge of the selection, *xterm* assumes you really meant the left edge, restores the original selection, then extends/contracts the left edge of the selection. Extension starts in the selection unit mode that the last selection or extension was performed in; you can multiple-click to cycle through them.

By cutting and pasting pieces of text without trailing new lines, you can take text from several places in different windows and form a command to the shell, for example, or take output from a program and insert it into your favorite editor. Since the cut buffer is globally shared among different applications, you should regard it as a 'file' whose contents you know. The terminal emulator and other text programs should be treating it as if it were a text file, i.e., the text is delimited by new lines.

The scroll region displays the position and amount of text currently showing in the window (highlighted) relative to the amount of text actually saved. As more text is saved (up to the maximum), the size of the highlighted area decreases.

Clicking button one with the pointer in the scroll region moves the adjacent line to the top of the display window.

Clicking button three moves the top line of the display window down to the pointer position.

Clicking button two moves the display to a position in the saved text that corresponds to the pointer's position in the scrollbar.

Unlike the VT102 window, the Tektronix window dows not allow the copying of text. It does allow Tektronix GIN mode, and in this mode the cursor will change from an arrow to a cross. Pressing any key will send that key and the current coordinate of the cross cursor. Pressing button one, two, or three will return the letters 'l', 'm', and 'r', respectively. If the 'shift' key is pressed when a pointer button is pressed, the corresponding upper case letter is sent. To distinguish a pointer button from a key, the high bit of the character is set (but this is bit is normally stripped unless the terminal mode is RAW; see *tty*(4) for details).

MENUS

Xterm has four menus, named *mainMenu*, *vtMenu*, *fontMenu*, and *tekMenu*. Each menu pops up under the correct combinations of key and button presses. Most menus are divided into two section, separated by a horizontal line. The top portion contains various modes that can be altered. A check mark appears next to a mode that is currently active. Selecting one of these modes toggles its state. The bottom portion of the menu are command entries; selecting one of these performs the indicated function.

The **xterm** menu pops up when the ‘‘control’’ key and pointer button one are pressed in a window. The *mainMenu* contains items that apply to both the VT102 and Tektronix windows. The **Secure Keyboard** mode is be used when typing in passwords or other sensitive data in an unsecure environment; see **SECURITY** below. Notable entries in the command section of the menu are the **Continue**, **Suspend**, **Interrupt**, **Hangup**, **Terminate** and **Kill** which sends the SIGCONT, SIGTSTP, SIGINT, SIGHUP, SIGTERM and SIGKILL signals, respectively, to the process group of the process running under *xterm* (usually the shell). The **Continue** function is especially useful if the user has accidentally typed CTRL-Z, suspending the process.

The *vtMenu* sets various modes in the VT102 emulation, and is popped up when the ‘‘control’’ key and pointer button two are pressed in the VT102 window. In the command section of this menu, the soft reset entry will reset scroll regions. This can be convenient when some program has left the scroll regions set incorrectly (often a problem when using VMS or TOPS-20). The full reset entry will clear the screen, reset tabs to every eight columns, and reset the terminal modes (such as wrap and smooth scroll) to their initial states just after *xterm* has finished processing the command line options.

The *fontMenu* sets the font used in the VT102 window. In addition to the default font and a number of alternatives that are set with resources, the menu offers the font last specified by the Set Font escape sequence (see the document *Xterm Control Sequences*) and the current selection as a font name (if the PRIMARY selection is owned).

The *tekMenu* sets various modes in the Tektronix emulation, and is popped up when the ‘‘control’’ key and pointer button two are pressed in the Tektronix window. The current font size is checked in the modes section of the menu. The **PAGE** entry in the command section clears the Tektronix window.

SECURITY

X environments differ in their security consciousness. MIT servers, run under *xdm*, are capable of using a ‘‘magic cookie’’ authorization scheme that can provide a reasonable level of security for many people. If your server is only using a host-based mechanism to control access to the server (see *xhost(1)*), then if you enable access for a host and other users are also permitted to run clients on that same host, there is every possibility that someone can run an application that will use the basic services of the X protocol to snoop on your activities, potentially capturing a transcript of everything you type at the keyboard. This is of particular concern when you want to type in a password or other sensitive data. The best solution to this problem is to use a better authorization mechanism that host-based control, but a simple mechanism exists for protecting keyboard input in *xterm*.

The **xterm** menu (see **MENUS** above) contains a **Secure Keyboard** entry which, when enabled, ensures that all keyboard input is directed *only* to *xterm* (using the GrabKeyboard protocol request). When an application prompts you for a password (or other sensitive data), you can enable **Secure Keyboard** using the menu, type in the data, and then disable **Secure Keyboard** using the menu again. Only one X client at a time can secure the keyboard, so when you attempt to enable **Secure Keyboard** it may fail. In this case, the bell will sound. If the **Secure Keyboard** succeeds, the foreground and background colors will be exchanged (as if you selected the **Reverse Video** entry in the **Modes** menu); they will be exchanged again when you exit secure mode. If the colors do *not* switch, then you should be *very* suspicious that you are being spoofed. If the application you are running displays a prompt before asking for the password, it is safest to enter secure mode *before* the prompt gets displayed, and to make sure that the prompt gets displayed correctly (in the new colors), to minimize the probability of spoofing. You can also bring up the menu again and make sure that a check mark appears next to the entry.

Secure Keyboard mode will be disabled automatically if your xterm window becomes iconified (or otherwise unmapped), or if you start up a reparenting window manager (that places a title bar or other decoration around the window) while in **Secure Keyboard** mode. (This is a feature of the X protocol not easily overcome.) When this happens, the foreground and background colors will be switched back and the bell will sound in warning.

CHARACTER CLASSES

Clicking the middle mouse button twice in rapid succession will cause all characters of the same class (e.g. letters, white space, punctuation) to be selected. Since different people have different preferences for what should be selected (for example, should filenames be selected as a whole or only the separate subnames), the default mapping can be overridden through the use of the *charClass* (class *CharClass*) resource.

This resource is simply a list of *range*:*value* pairs where the range is either a single number or *low-high* in the range of 0 to 127, corresponding to the ASCII code for the character or characters to be set. The *value* is arbitrary, although the default table uses the character number of the first character occurring in the set.

The default table is

```
static int charClass[128] = {
/* NUL  SOH  STX  ETX  EOT  ENQ  ACK  BEL */
    32,   1,   1,   1,   1,   1,   1,   1,
/*  BS   HT   NL   VT   NP   CR   SO   SI */
     1,  32,   1,   1,   1,   1,   1,   1,
/* DLE  DC1  DC2  DC3  DC4  NAK  SYN  ETB */
     1,   1,   1,   1,   1,   1,   1,   1,
/* CAN   EM  SUB  ESC   FS   GS   RS   US */
     1,   1,   1,   1,   1,   1,   1,   1,
/*  SP    !    "    #    $    %    &    ' */
    32,  33,  34,  35,  36,  37,  38,  39,
/*   (    )    *    +    ,    -    .    / */
    40,  41,  42,  43,  44,  45,  46,  47,
/*   0    1    2    3    4    5    6    7 */
    48,  48,  48,  48,  48,  48,  48,  48,
/*   8    9    :    ;    <    =    >    ? */
    48,  48,  58,  59,  60,  61,  62,  63,
/*   @    A    B    C    D    E    F    G */
    64,  48,  48,  48,  48,  48,  48,  48,
/*   H    I    J    K    L    M    N    O */
    48,  48,  48,  48,  48,  48,  48,  48,
/*   P    Q    R    S    T    U    V    W */
    48,  48,  48,  48,  48,  48,  48,  48,
/*   X    Y    Z    [    \    ]    ^    _ */
    48,  48,  48,  91,  92,  93,  94,  48,
/*   `    a    b    c    d    e    f    g */
    96,  48,  48,  48,  48,  48,  48,  48,
/*   h    i    j    k    l    m    n    o */
    48,  48,  48,  48,  48,  48,  48,  48,
/*   p    q    r    s    t    u    v    w */
    48,  48,  48,  48,  48,  48,  48,  48,
/*   x    y    z    {    |    }    ~  DEL */
    48,  48,  48, 123, 124, 125, 126,   1};
```

For example, the string ''33:48,37:48,45-47:48,64:48'' indicates that the exclamation mark, percent sign, dash, period, slash, and ampersand characters should be treated the same way as characters and numbers. This is very useful for cutting and pasting electronic mailing addresses and filenames.

ACTIONS

It is possible to rebind keys (or sequences of keys) to arbitrary strings for input, by changing the translations for the vt100 or tek4014 widgets. Changing the translations for events other than key and button events is not expected, and will cause unpredictable behavior. The following actions are provided for using within the *vt100* or *tek4014* **translations** resources:

bell([*percent***])**

This action rings the keyboard bell at the specified percentage above or below the base volume.

ignore() This action ignores the event but checks for special pointer position escape sequences.

insert() This action inserts the character or string associated with the key that was pressed.

insert-seven-bit()
This action is a synonym for **insert()**

insert-eight-bit()
This action inserts an eight-bit (Meta) version of the character or string associated with the key that was pressed. The exact action depends on the value of the **eightBitInput** resource.

insert-selection(*sourcename* **[, ...])**
This action inserts the string found in the selection or cutbuffer indicated by *sourcename*. Sources are checked in the order given (case is significant) until one is found. Commonly-used selections include: *PRIMARY*, *SECONDARY*, and *CLIPBOARD*. Cut buffers are typically named *CUT_BUFFER0* through *CUT_BUFFER7*.

keymap(*name***)**
This action dynamically defines a new translation table whose resource name is *name* with the suffix *Keymap* (case is significant). The name *None* restores the original translation table.

popup-menu(*menuname***)**
This action displays the specified popup menu. Valid names (case is significant) include: *mainMenu*, *vtMenu*, *fontMenu*, and *tekMenu*.

secure() This action toggles the *Secure Keyboard* mode described in the section named **SECURITY**, and is invoked from the **securekbd** entry in *mainMenu*.

select-start()
This action begins text selection at the current pointer location. See the section on **POINTER USAGE** for information on making selections.

select-extend()
This action tracks the pointer and extends the selection. It should only be bound to Motion events.

select-end(*destname* **[, ...])**
This action puts the currently selected text into all of the selections or cutbuffers specified by *destname*.

select-cursor-start()
This action is similar to **select-start** except that it begins the selection at the current text cursor position.

select-cursor-end(*destname* **[, ...])**
This action is similar to **select-end** except that it should be used with **select-cursor-start**.

set-vt-font(*d/1/2/3/4/5/6/e/s* **[,***normalfont* **[,** *boldfont***]])**
This action sets the font or fonts currently being used in the VT102 window. The first argument is a single character that specifies the font to be used: *d* or *D* indicate the default font (the font initially used when *xterm* was started), *1* through *6* indicate the fonts specified by the *font1* through *font6* resources, *e* or *E* indicate the normal and bold fonts that have been set through escape codes (or specified as the second and third action arguments, respectively), and *s* or *S* indicate the font selection (as made by programs such as *xfontsel(1)*) indicated by the second action argument.

start-extend()
This action is similar to **select-start** except that the selection is extended to the current pointer location.

start-cursor-extend()
This action is similar to **select-extend** except that the selection is extended to the current text cursor position.

string(*string***)**
This action inserts the specified text string as if it had been typed. Quotation is necessary if the string contains whitespace or non-alphanumeric characters. If the string argument begins with the characters ‘‘0x’’, it is interpreted as a hex character constant.

scroll-back(*count* **[,***units***])**
This action scrolls the text window backward so that text that had previously scrolled off the top of the screen is now visible. The *count* argument indicates the number of *units* (which may be *page*, *halfpage*, *pixel*, or *line*) by which to scroll.

scroll-forw(*count* **[,***units***])**
This action scrolls is similar to **scroll-back** except that it scrolls the other direction.

allow-send-events(*on/off/toggle***)**
This action set or toggles the **allowSendEvents** resource and is also invoked by the **allowsends** entry in *mainMenu*.

set-logging(*on/off/toggle***)**
This action toggles the **logging** resource and is also invoked by the **logging** entry in *mainMenu*.

redraw()
This action redraws the window and is also invoked by the *redraw* entry in *mainMenu*.

send-signal(*signame***)**
This action sends the signal named by *signame* to the *xterm* subprocess (the shell or program specified with the *–e* command line option) and is also invoked by the **suspend**, **continue**, **interrupt**, **hangup**, **terminate**, and *kill* entries in *mainMenu*. Allowable signal names are (case is not significant): *tstp* (if supported by the operating system), *suspend* (same as *tstp*), *cont* (if supported by the operating system), *int*, *hup*, *term*, *quit*, *alrm*, *alarm* (same as *alrm*) and *kill*.

quit() This action sends a SIGHUP to the subprogram and exits. It is also invoked by the **quit** entry in *mainMenu*.

set-scrollbar(*on/off/toggle***)**
This action toggles the **scrollbar** resource and is also invoked by the **scrollbar** entry in *vtMenu*.

set-jumpscroll(*on/off/toggle***)**
This action toggles the **jumpscroll** resource and is also invoked by the **jumpscroll** entry in *vtMenu*.

set-reverse-video(*on/off/toggle***)**
This action toggles the *reverseVideo* resource and is also invoked by the **reversevideo** entry in *vtMenu*.

set-autowrap(*on/off/toggle***)**
This action toggles automatic wrapping of long lines and is also invoked by the **autowrap** entry in *vtMenu*.

set-reversewrap(*on/off/toggle***)**
This action toggles the **reverseWrap** resource and is also invoked by the **reversewrap** entry in *vtMenu*.

set-autolinefeed(*on/off/toggle***)**
This action toggles automatic insertion of linefeeds and is also invoked by the **autolinefeed** entry in *vtMenu*.

set-appcursor(*on/off/toggle***)**
This action toggles the handling Application Cursor Key mode and is also invoked by the Bappcursor entry in *vtMenu*.

set-appkeypad(*on/off/toggle***)**
This action toggles the handling of Application Keypad mode and is also invoked by the

appkeypad entry in *vtMenu*.

set-scroll-on-key(*on/off/toggle***)**
This action toggles the **scrollKey** resource and is also invoked from the **scrollkey** entry in *vtMenu*.

set-scroll-on-tty-output(*on/off/toggle***)**
This action toggles the **scrollTtyOutput** resource and is also invoked from the **scrollttyoutput** entry in *vtMenu*.

set-allow132(*on/off/toggle***)**
This action toggles the **c132** resource and is also invoked from the **allow132** entry in *vtMenu*.

set-cursesemul(*on/off/toggle***)**
This action toggles the **curses** resource and is also invoked from the **cursesemul** entry in *vtMenu*.

set-visual-bell(*on/off/toggle***)**
This action toggles the **visualBell** resource and is also invoked by the **visualbell** entry in *vtMenu*.

set-marginbell(*on/off/toggle***)**
This action toggles the **marginBell** resource and is also invoked from the **marginbell** entry in *vtMenu*.

set-altscreen(*on/off/toggle***)**
This action toggles between the alternate and current screens.

soft-reset()
This action resets the scrolling region and is also invoked from the **softreset** entry in *vtMenu*.

hard-reset()
This action resets the scrolling region, tabs, window size, and cursor keys and clears the screen. It is also invoked from the **hardreset** entry in *vtMenu*.

clear-saved-lines()
This action does **hard-reset()** (see above) and also clears the history of lines saved off the top of the screen. It is also invoked from the **clearsavedlines** entry in *vtMenu*.

set-terminal-type(*type***)**
This action directs output to either the *vt* or *tek* windows, according to the *type* string. It is also invoked by the **tekmode** entry in *vtMenu* and the **vtmode** entry in *tekMenu*.

set-visibility(*vt/tek***,***on/off/toggle***)**
This action controls whether or not the *vt* or *tek* windows are visible. It is also invoked from the **tekshow** and **vthide** entries in *vtMenu* and the **vtshow** and **tekhide** entries in *tekMenu*.

set-tek-text(*large/2/3/small***)**
This action sets font used in the Tektronix window to the value of the resources **tektextlarge**, **tektext2**, **tektext3**, and **tektextsmall** according to the argument. It is also by the entries of the same names as the resources in *tekMenu*.

tek-page()
This action clears the Tektronix window and is also invoked by the **tekpage** entry in *tekMenu*.

tek-reset()
This action resets the Tektronix window and is also invoked by the *tekreset* entry in *tekMenu*.

tek-copy()
This action copies the escape codes used to generate the current window contents to a file in the current directory beginning with the name COPY. It is also invoked from the *tekcopy* entry in *tekMenu*.

visual-bell()
This action flashes the window quickly.

The Tektronix window also has the following action:

gin-press(*l/L/m/M/r/R***)**
This action sends the indicated graphics input code.

The default bindings in the VT102 window are:

```
 Shift <KeyPress> Prior:        scroll-back(1,halfpage) \n\
  Shift <KeyPress> Next:        scroll-forw(1,halfpage) \n\
Shift <KeyPress> Select:        select-cursor-start() \
                                select-cursor-end(PRIMARY, CUT_BUFFER0) \n\
Shift <KeyPress> Insert:        insert-selection(PRIMARY, CUT_BUFFER0) \n\
       ˜Meta<KeyPress>:         insert-seven-bit() \n\
        Meta<KeyPress>:         insert-eight-bit() \n\
      !Ctrl <Btn1Down>:         popup-menu(mainMenu) \n\
 !Lock Ctrl <Btn1Down>:         popup-menu(mainMenu) \n\
      ˜Meta <Btn1Down>:         select-start() \n\
    ˜Meta <Btn1Motion>:         select-extend() \n\
      !Ctrl <Btn2Down>:         popup-menu(vtMenu) \n\
 !Lock Ctrl <Btn2Down>:         popup-menu(vtMenu) \n\
˜Ctrl ˜Meta <Btn2Down>:         ignore() \n\
  ˜Ctrl ˜Meta <Btn2Up>:         insert-selection(PRIMARY, CUT_BUFFER0) \n\
      !Ctrl <Btn3Down>:         popup-menu(fontMenu) \n\
 !Lock Ctrl <Btn3Down>:         popup-menu(fontMenu) \n\
˜Ctrl ˜Meta <Btn3Down>:         start-extend() \n\
    ˜Meta <Btn3Motion>:         select-extend() \n\
                                    <BtnUp>:select-end(PRIMARY, CUT_BUFFER0) \n\
            <BtnDown>:          bell(0)
```

The default bindings in the Tektronix window are:

```
      ˜Meta<KeyPress>:          insert-seven-bit() \n\
       Meta<KeyPress>:          insert-eight-bit() \n\
     !Ctrl <Btn1Down>:          popup-menu(mainMenu) \n\
!Lock Ctrl <Btn1Down>:          popup-menu(mainMenu) \n\
     !Ctrl <Btn2Down>:          popup-menu(tekMenu) \n\
!Lock Ctrl <Btn2Down>:          popup-menu(tekMenu) \n\
Shift ˜Meta<Btn1Down>:          gin-press(L) \n\
      ˜Meta<Btn1Down>:          gin-press(l) \n\
Shift ˜Meta<Btn2Down>:          gin-press(M) \n\
      ˜Meta<Btn2Down>:          gin-press(m) \n\
Shift ˜Meta<Btn3Down>:          gin-press(R) \n\
      ˜Meta<Btn3Down>:          gin-press(r)
```

Below is a sample how of the **keymap()** action is used to add special keys for entering commonly-typed works:

```
*VT100.Translations: #override <Key>F13: keymap(dbx)
*VT100.dbxKeymap.translations: \
        <Key>F14:       keymap(None) \n\
        <Key>F17:       string("next") string(0x0d) \n\
        <Key>F18:       string("step") string(0x0d) \n\
        <Key>F19:       string("continue") string(0x0d) \n\
        <Key>F20:       string("print ") insert-selection(PRIMARY, CUT_BUFFER0)
```

ENVIRONMENT

Xterm sets the environment variables ''TERM'' and ''TERMCAP'' properly for the size window you have created. It also uses and sets the environment variable ''DISPLAY'' to specify which bit map display terminal to use. The environment variable ''WINDOWID'' is set to the X window id number of the *xterm* window.

SEE ALSO

resize(1), X(1), pty(4), tty(4)
Xterm Control Sequences (in the *xterm* source directory)

BUGS

Large pastes do not work on some systems. This is not a bug in *xterm*; it is a bug in the pseudo terminal driver of those systems. *xterm* feeds large pastes to the pty only as fast as the pty will accept data, but some pty drivers do not return enough information to know if the write has succeeded.

Many of the options are not resettable after *xterm* starts.

The Tek widget does not support key/button re-binding.

Only fixed-width, character-cell fonts are supported.

This program still needs to be rewritten. It should be split into very modular sections, with the various emulators being completely separate widgets that don't know about each other. Ideally, you'd like to be able to pick and choose emulator widgets and stick them into a single control widget.

There needs to be a dialog box to allow entry of log file name and the COPY file name.

COPYRIGHT

AUTHORS

Far too many people, including:

Loretta Guarino Reid (DEC-UEG-WSL), Joel McCormack (DEC-UEG-WSL), Terry Weissman (DEC-UEG-WSL), Edward Moy (Berkeley), Ralph R. Swick (MIT-Athena), Mark Vandevoorde (MIT-Athena), Bob McNamara (DEC-MAD), Jim Gettys (MIT-Athena), Bob Scheifler (MIT X Consortium), Doug Mink (SAO), Steve Pitschke (Stellar), Ron Newman (MIT-Athena), Jim Fulton (MIT X Consortium), Dave Serisky (HP), Jonathan Kamens (MIT-Athena)

NAME

xwd - dump an image of an X window

SYNOPSIS

xwd [-debug] [-help] [-nobdrs] [-out *file*] [-xy] [-frame] [-add *value*] [-root | -id *id* | -name *name*] [-icmap] [-screen] [-display *display*]

DESCRIPTION

Xwd is an X Window System window dumping utility. *Xwd* allows X users to store window images in a specially formatted dump file. This file can then be read by various other X utilities for redisplay, printing, editing, formatting, archiving, image processing, etc. The target window is selected by clicking the pointer in the desired window. The keyboard bell is rung once at the beginning of the dump and twice when the dump is completed.

OPTIONS

-display *display*
This argument allows you to specify the server to connect to; see *X(1)*.

-help Print out the 'Usage:' command syntax summary.

-nobdrs This argument specifies that the window dump should not include the pixels that compose the X window border. This is useful in situations where you may wish to include the window contents in a document as an illustration.

-out *file* This argument allows the user to explicitly specify the output file on the command line. The default is to output to standard out.

-xy This option applies to color displays only. It selects 'XY' format dumping instead of the default 'Z' format.

-add *value*
This option specifies an signed value to be added to every pixel.

-frame This option indicates that the window manager frame should be included when manually selecting a window.

-root This option indicates that the root window should be selected for the window dump, without requiring the user to select a window with the pointer.

-id *id* This option indicates that the window with the specified resource id should be selected for the window dump, without requiring the user to select a window with the pointer.

-name *name*
This option indicates that the window with the specified WM_NAME property should be selected for the window dump, without requiring the user to select a window with the pointer.

-icmap Normally the colormap of the chosen window is used to obtain RGB values. This option forces the first installed colormap of the screen to be used instead.

-screen This option indicates that the GetImage request used to obtain the image should be done on the root window, rather than directly on the specified window. In this way, you can obtain pieces of other windows that overlap the specified window, and more importantly, you can capture menus or other popups that are independent windows but appear over the specified window.

ENVIRONMENT

DISPLAY
To get default host and display number.

FILES

XWDFile.h
X Window Dump File format definition file.

SEE ALSO

xwud(1), xpr(1), X(1)

COPYRIGHT

AUTHORS

Tony Della Fera, Digital Equipment Corp., MIT Project Athena
William F. Wyatt, Smithsonian Astrophysical Observatory

NAME

xwininfo – window information utility for X

SYNOPSIS

xwininfo [–help] [–id *id*] [–root] [–name *name*] [–int] [–children] [–tree] [–stats] [–bits] [–events] [–size] [–wm] [–shape] [–frame] [–all] [–english] [–metric] [–display *display*]

DESCRIPTION

Xwininfo is a utility for displaying information about windows. Various information is displayed depending on which options are selected. If no options are chosen, **–stats** is assumed.

The user has the option of selecting the target window with the mouse (by clicking any mouse button in the desired window) or by specifying its window id on the command line with the **–id** option. Or instead of specifying the window by its id number, the **–name** option may be used to specify which window is desired by name. There is also a special **–root** option to quickly obtain information on the screen's root window.

OPTIONS

–help Print out the 'Usage:' command syntax summary.

–id *id* This option allows the user to specify a target window *id* on the command line rather than using the mouse to select the target window. This is very useful in debugging X applications where the target window is not mapped to the screen or where the use of the mouse might be impossible or interfere with the application.

–name *name*
This option allows the user to specify that the window named *name* is the target window on the command line rather than using the mouse to select the target window.

–root This option specifies that X's root window is the target window. This is useful in situations where the root window is completely obscured.

–int This option specifies that all X window ids should be displayed as integer values. The default is to display them as hexadecimal values.

–children
This option causes the root, parent, and children windows' ids and names of the selected window to be displayed.

–tree This option is like **–children** but displays all children recursively.

–stats This option causes the display of various attributes pertaining to the location and appearance of the selected window. Information displayed includes the location of the window, its width and height, its depth, border width, class, colormap id if any, map state, backing-store hint, and location of the corners.

–bits This option causes the display of various attributes pertaining to the selected window's raw bits and how the selected window is to be stored. Displayed information includes the selected window's bit gravity, window gravity, backing-store hint, backing-planes value, backing pixel, and whether or not the window has save-under set.

–events This option causes the selected window's event masks to be displayed. Both the event mask of events wanted by some client and the event mask of events not to propagate are displayed.

–size This option causes the selected window's sizing hints to be displayed. Displayed information includes: for both the normal size hints and the zoom size hints, the user supplied location if any; the program supplied location if any; the user supplied size if any; the program supplied size if any; the minimum size if any; the maximum size if any; the resize increments if any; and the minimum and maximum aspect ratios if any.

–wm This option causes the selected window's window manager hints to be displayed. Information displayed may include whether or not the application accepts input, what the window's icon window # and name is, where the window's icon should go, and what the window's initial state should be.

–shape This option causes the selected window's window and border shape extents to be displayed.

–frame This option causes window manager frames to be considered when manually selecting windows.

–metric This option causes all individual height, width, and x and y positions to be displayed in millimeters as well as number of pixels, based on what the server thinks the resolution is. Geometry specifications that are in **+x+y** form are not changed.

–english This option causes all individual height, width, and x and y positions to be displayed in inches (and feet, yards, and miles if necessary) as well as number of pixels. **–metric** and **–english** may both be enabled at the same time.

–all This option is a quick way to ask for all information possible.

–display *display*
This option allows you to specify the server to connect to; see *X(1)*.

EXAMPLE

The following is a sample summary taken with no options specified:

```
xwininfo: Window id: 0x60000f "xterm"

  Absolute upper-left X: 2
  Absolute upper-left Y: 85
  Relative upper-left X:  0
  Relative upper-left Y:  25
  Width: 579
  Height: 316
  Depth: 8
  Visual Class: PseudoColor
  Border width: 0
  Class: InputOutput
  Colormap: 0x27 (installed)
  Bit Gravity State: NorthWestGravity
  Window Gravity State: NorthWestGravity
  Backing Store State: NotUseful
  Save Under State: no
  Map State: IsViewable
  Override Redirect State: no
  Corners:  +2+85  -699+85  -699-623  +2-623
  -geometry 80x24+0+58
```

ENVIRONMENT

DISPLAY
To get the default host and display number.

SEE ALSO

X(1), xprop(1)

BUGS

Using **–stats –bits** shows some redundant information.

The -geometry string displayed must make assumptions about the window's border width and the behavior of the application and the window manager. As a result, the location given is not always correct.

COPYRIGHT

AUTHOR

Mark Lillibridge, MIT Project Athena

NAME

xwud - image displayer for X

SYNOPSIS

xwud [-in *file*] [-noclick] [-geometry *geom*] [-display *display*] [-new] [-std <maptype>] [-raw] [-vis <vis-type-or-id>] [-help] [-rv] [-plane *number*] [-fg *color*] [-bg *color*]

DESCRIPTION

Xwud is an X Window System image undumping utility. *Xwud* allows X users to display in a window an image saved in a specially formatted dump file, such as produced by *xwd(1)*.

OPTIONS

-bg *color*
If a bitmap image (or a single plane of an image) is displayed, this option can be used to specify the color to display for the "0" bits in the image.

-display *display*
This option allows you to specify the server to connect to; see *X(1)*.

-fg *color*
If a bitmap image (or a single plane of an image) is displayed, this option can be used to specify the color to display for the "1" bits in the image.

-geometry *geom*
This option allows you to specify the size and position of the window. Typically you will only want to specify the position, and let the size default to the actual size of the image.

-help
Print out a short description of the allowable options.

-in *file*
This option allows the user to explicitly specify the input file on the command line. If no input file is given, the standard input is assumed.

-new
This option forces creation of a new colormap for displaying the image. If the image characteristics happen to match those of the display, this can get the image on the screen faster, but at the cost of using a new colormap (which on most displays will cause other windows to go technicolor).

-noclick
Clicking any button in the window will terminate the application, unless this option is specified. Termination can always be achieved by typing 'q', 'Q', or ctrl-c.

-plane *number*
You can select a single bit plane of the image to display with this option. Planes are numbered with zero being the least significant bit. This option can be used to figure out which plane to pass to *xpr(1)* for printing.

-raw
This option forces the image to be displayed with whatever color values happen to currently exist on the screen. This option is mostly useful when undumping an image back onto the same screen that the image originally came from, while the original windows are still on the screen, and results in getting the image on the screen faster.

-rv
If a bitmap image (or a single plane of an image) is displayed, this option forces the foreground and background colors to be swapped. This may be needed when displaying a bitmap image which has the color sense of pixel values "0" and "1" reversed from what they are on your display.

-std *maptype*
This option causes the image to be displayed using the specified Standard Colormap. The property name is obtained by converting the type to upper case, prepending "RGB_", and appending "_MAP". Typical types are "best", "default", and "gray". See *xstdcmap(1)* for one way of creating Standard Colormaps.

-vis *vis-type-or-id*
This option allows you to specify a particular visual or visual class. The default is to pick the

"best" one. A particular class can be specified: "StaticGray", "GrayScale", "StaticColor", "PseudoColor", "DirectColor", or "TrueColor". Or "Match" can be specified, meaning use the same class as the source image. Alternatively, an exact visual id (specific to the server) can be specified, either as a hexadecimal number (prefixed with "0x") or as a decimal number. Finally, "default" can be specified, meaning to use the same class as the colormap of the root window. Case is not significant in any of these strings.

ENVIRONMENT

DISPLAY
To get default display.

FILES

XWDFile.h
X Window Dump File format definition file.

SEE ALSO

xwd(1), xpr(1), xstdcmap(1), X(1)

COPYRIGHT

AUTHOR

Bob Scheifler, MIT X Consortium

NAME

Xserver – X Window System server

SYNOPSIS

X [:displaynumber] [–option ...] [ttyname]

DESCRIPTION

X is the generic name for the X Window System server. It is frequently a link or a copy of the appropriate server binary for driving the most frequently used server on a given machine.

STARTING THE SERVER

The server is usually started from the X Display Manager program *xdm*. This utility is run from the system boot files and takes care of keeping the server running, prompting for usernames and passwords, and starting up the user sessions. It is easily configured for sites that wish to provide nice, consistent interfaces for novice users (loading convenient sets of resources, starting up a window manager, clock, and nice selection of terminal emulator windows).

Installations that run more than one window system will still need to use the *xinit* utility. However, *xinit* is to be considered a tool for building startup scripts and is not intended for use by end users. Site administrators are **strongly** urged to use *xdm*, or build other interfaces for novice users.

When the X server starts up, it takes over the display. If you are running on a workstation whose console is the display, you cannot log into the console while the server is running.

NETWORK CONNECTIONS

The X server supports connections made using the following reliable byte-streams:

TCP/IP
The server listens on port 6000+*n*, where *n* is the display number.

Unix Domain
The X server uses */tmp/.X11-unix/X***n** as the filename for the socket, where *n* is the display number.

DECnet
The server responds to connections to object *X$X***n**, where *n* is the display number. This is not supported in all environments.

OPTIONS

All of the X servers accept the following command line options:

–a *number*
sets pointer acceleration (i.e. the ratio of how much is reported to how much the user actually moved the pointer).

–auth *authorization-file*
Specifies a file which contains a collection of authorization records used to authenticate access. See also the *xdm* and *Xsecurity* manual pages.

bc disables certain kinds of error checking, for bug compatibility with previous releases (e.g., to work around bugs in R2 and R3 xterms and toolkits). Deprecated.

–bs disables backing store support on all screens.

–c turns off key-click.

c *volume* sets key-click volume (allowable range: 0-100).

–cc *class* sets the visual class for the root window of color screens. The class numbers are as specified in the X protocol. Not obeyed by all servers.

–co *filename*
sets name of RGB color database.

–dpi *resolution*
sets the resolution of the screen, in dots per inch. To be used when the server cannot determine

the screen size from the hardware.

–f *volume*
sets feep (bell) volume (allowable range: 0-100).

–fc *cursorFont*
sets default cursor font.

–fn *font* sets the default font.

–fp *fontPath*
sets the search path for fonts. This path is a comma separated list of directories which the X server searches for font databases.

–help prints a usage message.

–I causes all remaining command line arguments to be ignored.

–ld *kilobytes*
sets the data space limit of the server to the specified number of kilobytes. A value of zero makes the data size as large as possible. The default value of –1 leaves the data space limit unchanged. This option is not available in all operating systems.

–lf *files* sets the number-of-open-files limit of the server to the specified number. A value is zero makes the limit as large as possible. The default value of –1 leaves the limit unchanged. This option is not available in all operating systems.

–ls *kilobytes*
sets the stack space limit of the server to the specified number of kilobytes. A value of zero makes the stack size as large as possible. The default value of –1 leaves the stack space limit unchanged. This option is not available in all operating systems.

–logo turns on the X Window System logo display in the screen-saver. There is currently no way to change this from a client.

nologo turns off the X Window System logo display in the screen-saver. There is currently no way to change this from a client.

–p *minutes*
sets screen-saver pattern cycle time in minutes.

–r turns off auto-repeat.

r turns on auto-repeat.

–s *minutes*
sets screen-saver timeout time in minutes.

–su disables save under support on all screens.

–t *number*
sets pointer acceleration threshold in pixels (i.e. after how many pixels pointer acceleration should take effect).

–to *seconds*
sets default connection timeout in seconds.

tty*xx* ignored, for servers started the ancient way (from init).

v sets video-off screen-saver preference.

–v sets video-on screen-saver preference.

–wm forces the default backing-store of all windows to be WhenMapped; a cheap trick way of getting backing-store to apply to all windows.

–x *extension*

loads the specified extension at init. Not supported in most implementations.

You can also have the X server connect to *xdm* using XDMCP. Although this is not typically useful as it does not allow *xdm* to manage the server process, it can be used to debug XDMCP implementations, and serves as a sample implementation of the server side of XDMCP. For more information on this protocol, see the *X Display Manager Control Protocol* specification. The following options control the behavior of XDMCP.

–query *host-name*
Enable XDMCP and send Query packets to the specified host.

–broadcast
Enable XDMCP and broadcast BroadcastQuery packets to the network. The first responding display manager will be chosen for the session.

–indirect *host-name*
Enable XDMCP and send IndirectQuery packets to the specified host.

–port *port-num*
Use an alternate port number for XDMCP packets. Must be specified before any –query, –broadcast or –indirect options.

–once Normally, the server keeps starting sessions, one after the other. This option makes the server exit after the first session is over.

–class *display-class*
XDMCP has an additional display qualifier used in resource lookup for display-specific options. This option sets that value, by default it is "MIT-Unspecified" (not a very useful value).

–cookie *xdm-auth-bits*
When testing XDM-AUTHENTICATION-1, a private key is shared between the server and the manager. This option sets the value of that private data (not that it is very private, being on the command line!).

–displayID *display-id*
Yet another XDMCP specific value, this one allows the display manager to identify each display so that it can locate the shared key.

Many servers also have device-specific command line options. See the manual pages for the individual servers for more details.

SECURITY

The X server implements a simplistic authorization protocol, MIT-MAGIC-COOKIE-1 which uses data private to authorized clients and the server. This is a rather trivial scheme; if the client passes authorization data which is the same as the server has, it is allowed access. This scheme is worse than the host-based access control mechanisms in environments with unsecure networks as it allows any host to connect, given that it has discovered the private key. But in many environments, this level of security is better than the host-based scheme as it allows access control per-user instead of per-host.

In addition, the server provides support for a DES-based authorization scheme, XDM-AUTHORIZATION-1, which is more secure (given a secure key distribution mechanism). This authorization scheme can be used in conjunction with XDMCP's authentication scheme (XDM-AUTHENTICATION-1) or in isolation.

The authorization data is passed to the server in a private file named with the **–auth** command line option. Each time the server is about to accept the first connection after a reset (or when the server is starting), it reads this file. If this file contains any authorization records, the local host is not automatically allowed access to the server, and only clients which send one of the authorization records contained in the file in the connection setup information will be allowed access. See the *Xau* manual page for a description of the binary format of this file. Maintenance of this file, and distribution of its contents to remote sites for use there is left as an exercise for the reader.

The server also provides support for SUN-DES-1, using Sun's Secure RPC. It involves encrypting data with the X server's public key. See the *Xsecurity* manual page for more information.

The X server also uses a host-based access control list for deciding whether or not to accept connections from clients on a particular machine. If no other authorization mechanism is being used, this list initially consists of the host on which the server is running as well as any machines listed in the file */etc/X***n***.hosts*, where **n** is the display number of the server. Each line of the file should contain either an Internet hostname (e.g. expo.lcs.mit.edu) or a DECnet hostname in double colon format (e.g. hydra::). There should be no leading or trailing spaces on any lines. For example:

```
joesworkstation
corporate.company.com
star::
bigcpu::
```

Users can add or remove hosts from this list and enable or disable access control using the *xhost* command from the same machine as the server.

The X protocol intrinsically does not have any notion of window operation permissions or place any restrictions on what a client can do; if a program can connect to a display, it has full run of the screen. Sites that have better authentication and authorization systems (such as Kerberos) might wish to make use of the hooks in the libraries and the server to provide additional security models.

SIGNALS

The X server attaches special meaning to the following signals:

SIGHUP This signal causes the server to close all existing connections, free all resources, and restore all defaults. It is sent by the display manager whenever the main user's main application (usually an *xterm* or window manager) exits to force the server to clean up and prepare for the next user.

SIGTERM
This signal causes the server to exit cleanly.

SIGUSR1
This signal is used quite differently from either of the above. When the server starts, it checks to see if it has inherited SIGUSR1 as SIG_IGN instead of the usual SIG_DFL. In this case, the server sends a SIGUSR1 to its parent process after it has set up the various connection schemes. *Xdm* uses this feature to recognize when connecting to the server is possible.

FONTS

Fonts are usually stored as individual files in directories. The X server can obtain fonts from directories and/or from font servers. The list of directories and font servers the X server uses when trying to open a font is controlled by the *font path*. Although most sites will choose to have the X server start up with the appropriate font path (using the *–fp* option mentioned above), it can be overridden using the *xset* program.

The default font path for the X server contains four directories:

/usr/lib/X11/fonts/misc
This directory contains many miscellaneous bitmap fonts that are useful on all systems. It contains a family of fixed-width fonts, a family of fixed-width fonts from Dale Schumacher, several Kana fonts from Sony Corporation, two JIS Kanji fonts, two Hangul fonts from Daewoo Electronics, two Hebrew fonts from Joseph Friedman, the standard cursor font, two cursor fonts from Digital Equipment Corporation, and cursor and glyph fonts from Sun Microsystems. It also has various font name aliases for the fonts, including **fixed** and **variable**.

/usr/lib/X11/fonts/Speedo
This directory contains outline fonts for Bitstream's Speedo rasterizer. A single font face, in normal, bold, italic, and bold italic, is provided, contributed by Bitstream, Inc.

/usr/lib/X11/fonts/75dpi
This directory contains bitmap fonts contributed by Adobe Systems, Inc., Digital Equipment

Corporation, Bitstream, Inc., Bigelow and Holmes, and Sun Microsystems, Inc. for 75 dots per inch displays. An integrated selection of sizes, styles, and weights are provided for each family.

/usr/lib/X11/fonts/100dpi
This directory contains 100 dots per inch versions of some of the fonts in the *75dpi* directory.

Font databases are created by running the *mkfontdir* program in the directory containing the compiled versions of the fonts (the *.pcf* files). Whenever fonts are added to a directory, *mkfontdir* should be rerun so that the server can find the new fonts. **If *mkfontdir* is not run, the server will not be able to find any fonts in the directory.**

DIAGNOSTICS

Too numerous to list them all. If run from *init(8)*, errors are typically logged in the file */usr/adm/X*msgs*,

FILES

/etc/X*.hosts	Initial access control list
/usr/lib/X11/fonts/misc, /usr/lib/X11/fonts/75dpi, /usr/lib/X11/fonts/100dpi	Bitmap font directories
/usr/lib/X11/fonts/Speedo	Outline font directories
/usr/lib/X11/fonts/PEX	PEX font directories
/usr/lib/X11/rgb.txt	Color database
/tmp/.X11-unix/X*	Unix domain socket
/usr/adm/X*msgs	Error log file

SEE ALSO

X(1), bdftopcf(1), mkfontdir(1), fs(1), xauth(1), xdm(1), xhost(1), xinit(1), xset(1), xsetroot(1), xterm(1), Xdec(1), Xibm(1), XmacII(1), Xmips(1), Xqdss(1), Xqvss(1), Xsun(1), Xtek(1), X386(1) *X Window System Protocol, Definition of the Porting Layer for the X v11 Sample Server, Strategies for Porting the X v11 Sample Server, Godzilla's Guide to Porting the X V11 Sample Server*

BUGS

The option syntax is inconsistent with itself and *xset(1)*.

The acceleration option should take a numerator and a denominator like the protocol.

If *X* dies before its clients, new clients won't be able to connect until all existing connections have their TCP TIME_WAIT timers expire.

The color database is missing a large number of colors.

COPYRIGHT

AUTHORS

The sample server was originally written by Susan Angebranndt, Raymond Drewry, Philip Karlton, and Todd Newman, from Digital Equipment Corporation, with support from a large cast. It has since been extensively rewritten by Keith Packard and Bob Scheifler, from MIT.

NAME

X386 - X Window System server for UNIX System V/386

SYNOPSIS

X386 [:displaynumber] [option] ...

DESCRIPTION

X386 is the generic sample server for the X Window System, Version 11 Release 5 on i386 hardware. It is normally started by the *xdm(1)* daemon or by a script that runs the program *xinit(1)* such as *startx*.

New extensions supported by *X386* in X11R5 include X3D-PEX (PHIGS/+ Extension to X) and XTestExtension1 (Input Synthesis Extension).

CONFIGURATIONS

X386 operates under ISC, SCO, ESIX & AT&T UNIX System V/3.2 and UHC, ESIX, Dell & AT&T UNIX System V/4.0. The server supports the following popular SuperVGA boards in 256 color mode:

SuperVGA Cards	Max Res	ChipSet
Compuadd Hi-Rez card w/1meg	1024x768	ET4000
Diamond SpeedStar	1024x768	ET4000
EIZO MD-10	800x600	ET3000
GENOA 5300/5400	800x600	ET3000
GENOA 6400	800x600	GVGA
Optima Mega/1024	1024x768	ET4000
Orchid ProDesigner	800x600	ET3000
Orchid ProDesigner II/1024	1024x768	ET4000
Paradise VGA Professional	640x480	PVGA1A
Paradise VGA 1024	640x480	WD90C00
Sigma Legend	1024x768	ET4000
STB PowerGraph w/1meg	1024x768	ET4000
Swan SVGA equipped with VCO chip	1024x768	ET4000
TRICOM Mega/1024	1024x768	ET4000

The core X11R5 source tree as supplied in the public release was compiled and tested under ISC 2.2.1 (UNIX System V/3.2), Dell and AT&T UNIX System V/4.0.

NETWORK CONNECTIONS

X386 supports connections made using the following reliable byte-streams:

Unix Domain

X386 uses */tmp/.X11-unix/X***n** as the filename for the socket, where *n* is the display number, (SVR4).

TCP/IP

X386 listens on port htons(6000+*n*), where *n* is the display number, (SVR3 and SVR4).

OPTIONS

See the command line switches described in the *Xserver(1)* manual page. *X386* does not accept any additional switches.

KEYBOARD

Multiple key presses recognized directly by *X386* are:

Ctrl+Alt+Backspace

Immediately kills the server -- no questions asked. (Can be disabled by specifying "dontzap" in the configuration file.)

Ctrl+Alt+Keypad-Plus

Change video mode to next one specified in the configuration file, (increasing video resolution order).

Ctrl+Alt+Keypad-Minus
Change video mode to previous one specified in the configuration file, (decreasing video resolution order).

SETUP

X386 uses the configuration file /usr/X386/Xconfig for its initial setup. This file is composed of the following sections:

General Server Parameters
Keyboard Configuration
Mouse Configuration
Video Mode Setup
Table of Known Video Modes

The **General Server Parameters** section lets you adjust some generic server parameter which seldom change for a given site. (The **fontpath** parameter can also be set from the command line, see *Xserver(1)*.):

fontpath sets the search path for fonts. This path is a comma separated list of directories which the sample server searches for font databases.

rgbpath sets the name of RGB color database.

The **Keyboard Configuration** section starts with a keyword describing which kind of driver should be used: **keyboard** (the normal device) or **xqueue** (the eventque driver). Note that the latter one exists only for compatibility with older releases and shouldn't be used. Following this keyword the following options can be specified:

autorepeat *delay rate*
changes the behavior of the autorepeat of the keyboard.

dontzap disallows the use of the **Ctrl+Alt+Backspace** sequence. This sequence allows you to terminate the server.

servernum
forces the server to handle the numlock key internally. The server sends virtual key-events so applications can use the numberpad.

xleds *led* **...**
makes *led* available for clients instead of using the traditional function (Scroll Lock, Caps Lock & Num Lock)

Similar to above, the **Mouse Configuration** section starts with a special keyword, but here we must select the type of mouse (i.e. it's protocol) that is connected to the computer. (Using the *xqueue* driver this section is obsolete since mouse I/O is handled by the xqueue-driver.) The mouse types available are:

busmouse
logitech
microsoft
mmseries
mouseman
mousesystems

(One should specify **busmouse** for the Logitech bus mouse.)

The following options can be specified after this keyword:

baudrate *rate*
sets the baudrate of the serial mouse to *rate*. For mice that allow dynamic speed adjustments (like logitech) the baudrate is changed in the mouse. Otherwise the rate is simply set on the

computer's side to allow mice with non-standard rates.

emulate3buttons
enables the emulation of the third mouse button for mice which only have two physical buttons. The third button is emulated by pressing both buttons simultaneously.

samplerate *rate*
sets the number of motion/button-events the mouse sends per second. This is currently only supported for logitech mice.

The **Graphics Driver Setup** section starts with the keyword **vga256**. After this keyword a variety of options may be specified:

staticgray,grayscale,staticcolor,pseudocolor,truecolor,directcolor
sets the visual class for the root window of the screen.

chipset *"name"*
specifies a chipset so the correct driver can be used. Possible chipsets are:

et3000
et4000
gvga
pvga

clocks *clock* **...**
specifies the dotclocks that are on your graphics board.

displaysize *xdim ydim*
sets the display size (internal) to *xdim* x *ydim* (measured in mm).

modes *"mode"* **...**
selects the display modes for this screen. The first one in the list will be the default display mode for startup. Internally this list is converted into a circular list. With **Ctrl+Alt+Keypad-Plus** and **Ctrl+Alt+Keypad-Minus** the current display mode may be changed, and the next (upward/downward) entry in the list will be used.

vendor *"vendorstring"*
allows the server to select a special behavior for *special* hardware. Currently the following strings are recognized:

legend

videoram *mem*
specifies the amount of videoram that is installed on the graphics board. This is measured in kBytes.

viewport *x0 y0*
sets the upper left corner of the initial display. If the virtual resolution is larger than what is physically displayed, then the initial display will be centered if **viewport** is not specified.

virtual *xdim ydim*
sets the virtual resolution. For example one might use a display with 800x600, but a virtual resolution of 1152x900. If the mouse touches the borders of the display the image scrolls accordingly. This is called panning.

Note that *X386* has some internal capabilities to determine what hardware it is running on. Thus normally the keywords *chipset*, *clocks*, *vendor* and *videoram* don't have to be specified. But there may be occasions when this autodetection mechanism fails, (for example, too high of load on the machine when you start the server). For cases like this, one should first run *X386* on an unloaded machine, look at the results of the autodetection (that are printed out during server startup) and then explicitly specify these parameters in the configuration file.

The last section is the **Table of Video Modes** which starts with the keyword **modedb**. A list of possible mode-records follows this keyword. For every mode string in the previously specified **modes** line, the server scans the list of mode-records looking for a matching *modename*. When a match occurs, then the *dotclock* value is compared to those found on the graphics board. If the *dotclock* value matches, then the rest of the values in the mode-record will be used for that display mode. If there is more than one matching mode-record (for a given *modename* and *dotclock*), then the last one matched will be used.

A mode-record consists of seven parts:

modename
: is the string that identifies this mode. If one is not specified, then the *modename* string of the last valid mode-record is used.

dotclock
: is the dotclock this mode uses. This is the basic timer for all video signals.

hdisp, hsyncstart, hsyncend, htotal
: is the horizontal timing

vdisp, vsyncstart, vsyncend, vtotal
: is the vertical timing

interlace
: sets interlace display mode.

+hsync, -hsync
: selects polarity of HSYNC signal

+vsync, -vsync
: selects polarity of VSYNC signal

For details on how to build your own video modes please refer to the tutorial written by Chin Fang.

A sample config file might look like this:

```
RGBPath      "/usr/X386/lib/X11/rgb"
FontPath     "/usr/X386/lib/X11/fonts/misc/,:zok:7000"

Keyboard
  AutoRepeat 500 5
  Xleds      1 2 3
  ServerNumLock
  DontZap

Logitech                              "/dev/tty00"
  BaudRate                            9600
  SampleRate                          150
  Emulate3Buttons

vga256
  Chipset     "et4000"
  Vendor      "legend"
  Clocks      25 28 36 62
  Virtual     1152 900
  ViewPort    0 0
  Modes       "1024x768" "640x480"

ModeDB
 "640x480"   25    640  672  768  800    480  490  492  525
```

28 640 672 768 800 480 490 492 525
"1024x768" 62 1024 1092 1220 1344 768 786 791 810

Note that a new feature of X11R5 is the ability of the X server to request fonts from a *font server*. One specifies a *font server* by placing a ":<hostname>:<tcp_port_number>" into the fontpath. In the above example, the fontpath "/usr/X386/lib/X11/fonts/misc/,:zok:7000" tells *X386* to first try to locate the font in the local directory /usr/X386/lib/X11/fonts/misc. If that fails, then request the font from the *font server* running on machine zok listening for connections on TCP port number 7000.

FILES

/usr/X386/bin/X386	The X server
/usr/X386/lib/X11/Xconfig	Server configuration file
/usr/X386/lib/X11/etc	Additional *X386* support files
/usr/X386/bin/*	Client binaries
/usr/X386/include/*	Header files
/usr/X386/lib/*	Libraries
/usr/X386/lib/X11/fonts/*	Fonts
/usr/X386/lib/X11/rgb.{dir,pag,txt}	Color names to RGB mapping
/usr/X386/lib/X11/XErrorDB	Client error message database
/usr/X386//lib/X11/app-defaults/*	Client resource specifications
/usr/X386/man/man?/*	Manual pages
/etc/X0.hosts	Initial access control list

SEE ALSO

X(1), Xserver(1), xdm(1), xinit(1)

BUGS

Starting clients while on another virtual screen may destroy the contents of the original screen or may even kill the server. The workaround is to use *xrefresh(1)* to refresh the screen.

AUTHORS

Thomas Roell, *roell@informatik.tu-muenchen.de*
TU-Muenchen: Server and SVR4 stuff

Mark W. Snitily, *mark@zok.sgcs.com*
SGCS: SVR3 support, X Consortium Sponsor

... and many more people out there on the net who helped with ideas and bug-fixes.

X386 X11R5 source and binaries are available from SGCS. Send email to *mark@zok.sgcs.com* or *...!mips!zok!mark* for details.

Name

Xdec – X server for Digital RISC machines

Syntax

Xdec [*options*]

Description

The command starts the X server. The command supports the following hardware configurations:

DECstation 2100 Monochrome or Color Workstations
DECstation 3100 Monochrome or Color Workstations
DECstation 5000/100/200 CX or MX Single or Multiscreen Workstations

This server should run on reasonable one bit or eight bit Ultrix/RISC TURBOchannel displays of any resolution, if correct device driver support is present. The server queries the device driver interface to determine if a suitable display device is installed, and if so, configures the server appropriately.

The command that executes the server is specified together with its command line options in the file or using xdm(1) and, therefore, is automatically run when your system is started in multiuser mode. Optionally, you can create an file containing device-dependent command line options (separated by spaces) and use it to start the server. Command line options specified in the command starting the X server override those specified in the file.

Start the server in bug compatibility mode (with the **bc** option) to remain bug-for-bug compatible with previous releases of the server.

Options

The following options are device independent. These options are available on all workstation hardware platforms running X11 servers.

–a *num*	Sets the mouse acceleration value to the specified number of pixels. The default is 4.
–auth *string*	Selects a file containing authorization records. Hosts listed in the file are allowed to access to the server's display. By default, no authorization file is used.
bc	Enables bug compatibility with previous releases.
–bs	Disables Backing Store. Backing Store is disabled if you specify the **–bs** option on the command line.
–c	Turns off key clicking.
c *num*	Sets the key click volume to the specified number (0-100). The default is 20.
–co *string*	Selects a color database. The default is
–f *num*	Sets the bell volume to the specified number (0-100). The default is 50.
–fc *string*	Sets the default cursor font. The default is
–fd *num*	Sets the font size in dots per inch. Acceptable values are 75 and 100.
–fn *string*	Sets the default text font. The default is
–fp *string*	Sets the default font directory path. The default font directory paths are:
+fp *path*[,*path*...]	Prepends the specified font paths to the default path.
fp+ *path*[,*path*...]	Appends the specified font paths to the default path.
–help	Displays the command usage message.
–I	Directs the server to ignore all remaining command line arguments.
–ld *num*	Limits data space to *num* Kbytes.
–ls *num*	Limits stack space to *num* Kbytes.

–logo	Enables the X logo screen saver.
logo	Disables the X logo screen saver.
–p *seconds*	Determines how long to wait before changing the background pattern of the screen saver. The default is 10 seconds.
–r	Turns off automatic repetition of keyboard key characters.
r	Turns on automatic repetition of keyboard key characters. The default is on.
–s *seconds*	Sets the screen-saver timeout value to the specified number of seconds. The default is 10.
–su	Disables Save-Unders.
–t *num*	Sets the mouse threshold in pixels. The mouse will accelerate only if the threshold is exceeded. The default threshold is 4.
–to *seconds*	Specifies the number of seconds for the server to wait before declaring a session connection timed out.
tty*xx*	Specifies the terminal on which the server is to start.
v	Specifies that a blank screen is used for the screen saver. This is the default screen saver.
–v	Specifies that a pattern (noblank) screen is used for the screen saver.
–wm	Enables the use of Backing Store with WhenMapped calls.
–x *string*	Loads the file *string* containing an X extension.
:*num*	On a multihead system, specifies the head on which the server is run (0 or 1). The default is 0.
–broadcast	Enables X Display Manager Control Protocol (XDMCP) and sends broadcast query packets over the network.
–displayID *display-id*	For XDMCP, identifies manufacturer's display ID for display
–indirect *hostname*	Enables XDMCP and sends indirect query packets to the specified host.
–once	Terminates the server after one session.
–port *num*	Specifies the UDP port number that will receive messages.
–query *hostname*	Enables XDMCP and sends query packets to the specified host.
–XDMCP_class *display-class*	Specifies an additional display qualifier used in resource management by XDMCP.

The following are device-dependent, vendor-specific options. When the server is run on multiscreen capable platforms, selected device-dependent options take an optional screen-specification argument. Omitting the screen-specification argument defines the parameter for all available screens.

–btn *num*	Specifies the number of buttons on the pointer device. The default is three for a mouse device and four for a tablet device.
–bp[*screen*] *color*	Sets the color of black pixels for the screen. The *color* argument can be a named color from the database or a number sign (#) followed by a hexidecimal number.
–class[*screen*] *visual class*	Sets the visual class for the root window of the screen. Possible values are and
–dpi[*screen*] *num*	Sets the dots per inch for the x and y coordinates.
–dpix[*screen*] *num*	Sets the dots per inch for the x coordinates.
–dpiy[*screen*] *num*	Sets the dots per inch for the y coordinates.

By default on multiscreen systems, the server presumes the left hand most screen is screen zero, screen one to its right, screen two yet further to its right, and so on. The cursor will track from the right hand edge of screen zero to the left hand edge of screen one, from the right hand edge of screen one to the left hand edge of screen two. This arrangement can be modified in various ways, to support other physical arrangements of monitors.

–edge_bottom*scr1 scr2* Attaches the bottom edge of the screen specified by *scr1* to the screen specified by *scr2*.

–edge_left*scr1 scr2* Attaches the left edge of the screen specified by *scr1* to the screen specified by *scr2*.

–edge_right*scr1 scr2* Attaches the right edge of the screen specified by *scr1* to the screen specified by *scr2*.

–edge_top*scr1 scr2* Attaches the top edge of the screen specified by *scr1* to the screen specified by *scr2*.

–wp[*screen*] *color* Sets the color of white pixels for the screen. The syntax for *color* is the same as for the argument to the **–bp** option.

–tb *n* Opens for graphics tablet communications.

–pcm *n* Opens for Protocol Control Module (PCM) communications. The two free serial ports on the DECstation correspond to and Dial boxes and button boxes must be connected through these two ports.

Restrictions

If options not listed in this reference page are used, the server may fail. Using invalid options for the X server in the file may cause the workstation to behave as if the X server is hung.

Multiscreen configurations can contain either two- or three-color frame buffer display devices or monochrome frame buffer display devices.

Color and monochrome frame buffer display devices can be installed in the same workstation, however applications built before X11 release 5 may become confused due to poor initial design of resource files.

To connect two screens, two command line options must be issued. Attaching two screens using only one argument produces a one-way mouse-travel path. You can create a wrap-around mouse path by attaching noncontiguous screen edges. The arguments are disabled on single screen systems.

Nonsensical screen connections are not allowed; the top edge of a particular screen must be connected with the bottom edge of another screen, and the right edge of a particular screen must be connected with the left edge of another screen. Left and right edges cannot be connected to top or bottom edges.

Examples

The following example specifies that screen has a resolution of 100x100 dots per inch and screen has a resolution of 75x70 dots per inch:

```
Xdec -dpi0 100 -dpix1 75 -dpiy1 70
```

If no screen is specified, the value specified is used for all screens. If the screen resolution is not specified using command line options, a default value based on pixel dimensions and screen size is calculated for each screen.

The following example specifies that black pixels on screen have the hexadecimal value 3a009e005c0 prefixed with a number sign (#) and white pixels on screen are color "wheat" from the X rgb color database.

```
Xdec -bp1 #3a009e005c0 -wp1 wheat
```

For monochrome display devices, values of 0 and 1 are the only valid pixel colors.

To specify the default visual class of a root window on a particular screen, append the screen number (0, 1, or 2) to the command line option. Possible visual classes are: StaticGray, StaticColor, PseudoColor, GrayScale, and TrueColor. The following example specifies that the screen root window is a TrueColor visual, and the screen root window is a PseudoColor visual.

Xdec -class0 TrueColor -class1 PseudoColor

The following example attaches screen above screen and screen to the right of screen (an L-shaped configuration):

Xdec -edge_top0 1 -edge_bottom1 0 -edge_right0 2 -edge_left2 0

The following example is identical to the default state (a horizontal line) with the addition of a wraparound from screen to screen

Xdec -edge_left0 2 -edge_right0 1 -edge_left1 0 -edge_right1 2 \
-edge_left2 1 -edge_right2 0

Files

See Also

X(1X), xdm(1), Xserver(1)

X Window System: The Complete Reference to Xlib, X Protocol, ICCCM, XLFD, by Robert W. Scheifler and James Gettys, Second Edition, Digital Press, 1990

"X Window System Toolkit: The Complete Programmer's Guide and Specification, by Paul J. Asente and Ralph R. Swick, Digital Press, 1990

OSF/MOTIF Programmer's Guide and OSF/MOTIF Reference Guide, Open Software Foundation, Prentice-Hall, 1990

NAME

XmacII – Macintosh II server for X Version 11

SYNOPSIS

XmacII [option] ...

DESCRIPTION

XmacII is the server for Version 11 of the X window system on Macintosh II hardware running A/UX. It will normally be started by the shell script *X11*.

CONFIGURATIONS

XmacII operates under A/UX Release 1.0 and later. Under A/UX Release 1.0, the server supports a single monochrome display. Color and multiple screens are supported under A/UX Release 1.1 and later.

OPTIONS

The following options can be given on the command line to the X server.

–screen *number* **–depth** *number*
: arranges that the given screen will operate at the given depth. Screens are numbered from 0 and correspond to the relative position of video cards in the Macintosh II slots. Supported depths are 1 (monochrome at 1 bit per pixel, the default value), and 8 (color at 8 bits per pixel, indexing a colormap with 256 entries). Multiple screens and color support require A/UX Release 1.1 or later.

–a *number*
: sets pointer acceleration (i.e. the ratio of how much is reported to how much the user actually moved the pointer).

–logo
: turns on the X Window System logo display in the screen-saver. There is currently no way to change this from a client.

nologo
: turns off the X Window System logo display in the screen-saver. There is currently no way to change this from a client.

–p *minutes*
: sets screen-saver pattern cycle time in minutes.

–r
: turns off auto-repeat.

r
: turns on auto-repeat.

–s *minutes*
: sets screen-saver timeout time in minutes.

–t *numbers*
: sets pointer acceleration threshold in pixels (i.e. after how many pixels pointer acceleration should take effect).

–to *seconds*
: sets default screensaver timeout in seconds.

–co *filename*
: sets name of RGB color database

–help
: prints a usage message

–fp *fontPath*
: sets the search path for fonts

–fc *cursorFont*
: sets default cursor font

–fn *font*
: sets the default font

BUTTON MAPPINGS

Many X clients assume the mouse has three buttons. The A/UX X server simulates the middle and right mouse buttons with keystrokes -- the left-arrow key generates middle button events, and the right-arrow key generates right button events -- the real mouse button generates left button events. The open-apple or cloverleaf key is the "Meta" modifier, Meta can also be obtained by pressing the up-arrow key. The down-arrow key duplicates the Control key. Meta, Control, and Shift are often used in combination with other keystrokes or mouse clicks. For example, the terminal emulator xterm pops up menus in response to control-left and control-middle. The original function of the arrow keys may be obtained by holding down the Option key while pressing one of the arrow keys.

SECURITY

X uses an access control list for deciding whether or not to accept a connection from a given client. This list initially consists of the machine on which the server is running, and any hosts listed in the file */etc/X0.hosts* This file should contain one line per host name, with no white space.

The user can manipulate a dynamic form of this list in the server using the *xhost(1)* program from the same machine as the server.

Unlike some window systems, *X* does not have any notion of window operation permissions or place any restrictions on what a client can do; if a program can connect to a display, it has full run of the screen. There is support for using authentication services on connection startup beyond the simple host name check, but it is not used in the standard distribution.

FONTS

Fonts are stored in individual files in the directory(ies) named by the font search path. These files may be created by the *fc* utility. The name of the file must correspond to the name of the font in lowercase with the suffix ".snf" appended. Font files may be stored in *compress*(1) format, in which case an additional suffix (usually ".Z", defined when the server is built) must be appended to the usual filename.

FILES

/usr/bin/X11/X11	Bourne shell script which starts the server and initial *xterm* client
$HOME/.x11start	Bourne shell script executed by initial *xterm* client which starts a slew of other clients. Reference copy in /usr/lib/X11/.x11start
/usr/bin/X11/XmacII	the server binary
/usr/bin/X11/Xrepair	ensures console viability after server exits
/usr/bin/X11/*	client binaries
/usr/lib/X11/fonts/*	
/usr/lib/X11/rgb.{dir,pag,txt}	color names to RGB mapping
/etc/X0.hosts	Initial access control list
/tmp/.X11-unix/X0	Unix domain socket
/usr/adm/X0msgs	Error log file
$HOME/.twmrc	customization for the *twm* window manager. Reference copy in /usr/lib/X11/Sample.twmrc
$HOME/.uwmrc	customization for the *uwm* window manager. Reference copy in /usr/lib/X11/default.uwmrc
/usr/lib/X11/XErrorDB	client error message database
/usr/lib/X11/app-defaults	client specific resource specifications
/usr/lib/X11/examples/Xaw/*	source code examples of applications built on the *X* Toolkit
/usr/lib/terminfo/x/xterm*	*terminfo* database entries descibing the terminal emulation capabilities of the *xterm* client. Reference copies in /usr/lib/X11/xterm*.tic

/usr/include/X11/bitmaps/∗icon   bitmap representations of icons used by the *twm* window manager

SEE ALSO

xinit(1), X(1)

BUGS

In the unlikely event the server crashes, it may leave the keyboard in a non-useful state. The X11 script executes the *Xrepair* command in an attempt to repair the damage.

COPYRIGHT

NAME

Xmips – Mips server for X Version 11

SYNOPSIS

Xmips [option] ...

DESCRIPTION

Xmips is the server for Version 11 of the X window system on Mips Computer hardware. It is normally started by *xdm (1)*, or *xinit (1)*.

CONFIGURATIONS

Xmips operates under RISC/os on workstation configuration systems. Under RISC/os you should use unix domain connections (:0 as host specifier) for local connections.

OPTIONS

-bp *color*	set default black-pixel color
-wp *color*	set default white-pixel color
-plock	process lock the server
-tlock	text lock the server
-dlock	data lock the server
-nice *incr*	change the scheduling priority of the server
-noreset	do not reset on last client close
-visual *PseudoColor|StaticColor|GrayScale|StaticGray*	use grayscale/color visual Note: The **-visual** option overrides the standard server option **-cc**.
-xled1	use led for xled1 instead of Scroll Lock
-xled2	use led for xled2 instead of Caps Lock
-xled3	use led for xled3 instead of Num Lock
-enable *color|mono*	Enable only the color or monochrome screen on a dual screen system. (By default the color screen will be screen 0, and the monochrome screen will be screen 1.)
-19inch	set color screen dpi for 19 inch monitor
-keyboard *Default|AT|XT|UNIX [unit#]*	selects type of keyboard to use
-pointer *Default|Mouseman [unit# [baud [sample_rate]]]*	Selects type of pointer to use. Optionally sets the baud rate and sampling rate. The baud rate may be set to 1200, 2400, 4800 and 9600. The baud rate should be left at 1200 on RS2030 systems. The sampling rate is a number greater than or equal to zero. Zero selects the fastest sampling rate available.

The server also takes all of the standard server options (See *Xserver(1)* for a complete list.)

SEE ALSO

Xserver(1), X(1), xinit(1), xdm(1)

NAME

Xqdss - VAXStation II/GPX server for the X Window System

DESCRIPTION

The QDSS hardware in the VAXStation II/GPX supports 8 plane or 4 plane displays.

OPTIONS

-bp *pixel-value* set default black-pixel color
-wp *pixel-value* set default white-pixel color

The server also takes all of the standard server options (See /fIXserver(1)/fP for a complete list.)

NOTE

You need to make sure your kernel is configured with DMA_BUFSIZ set to 10 in sys/data/qd_data.c, or things won't work right.

BUGS

Currently only numeric arguments from #000000 to #ffffff are accepted for default BlackPixel and WhitePixel, and no screen number is accepted.

The GPX hardware contains a polygon edge address generation engine which does not conform to the polygon edge model specified in the X11 protocol document. The effect is that GPX polygon edges extend on average one-half pixel farther out than they ought.

SEE ALSO

Xserver(1), xinit(1), X(1)

COPYRIGHT

AUTHORS

Donald Mullis and Matt Pappas, Digital Equipment Corporation.

Thanks to Ram Rao, Vasudev Bhandarkar and Mary Larsen of UEG for their 4-plane code and numerous bug fixes.

NAME

Xqvss - QVSS server for the X Window System

DESCRIPTION

This driver supports the QVSS and SM displays.

The QVSS hardware in the DEC VS2 is monochrome. The usable size is 864 lines by 1024 pixels/line. (Under some versions of Ultrix, the driver thinks the screen is 960 pixels wide; to fix this for a VR260 monitor, patch the kernel variable qv_def_scrn to 2 and rebuild the kernel.).

The SM hardware in the DEC VaxStation is functionally identical with the QVSS hardware, except for new cursor support and slightly different cabling.

Probably the most irritating problem with the QVSS hardware is the cursor support, which is limited to 16x16. As only ‘‘and’’ and ‘‘or’’ operations are available in the hardware, the driver inverts the cursor depending upon the pixels below the cursor. The server displays a reasonable cursor in most (but not all) circumstances. The SM hardware has better support — two planes instead of one; exactly what X needs.

Under Ultrix-32 release 1.1 and 4.2BSD, the window system will be slightly unreliable due to a kernel problem. Fixed in Ultrix32 release 1.2. Under Ultrix-32 1.1, your best bet is to use unix domain connections (unix:0 as host spec’s) for local connections.

OPTIONS

-bp *pixel-value* set default black-pixel color
-wp *pixel-value* set default white-pixel color

The server also takes all of the standard server options (See *Xserver(1)* for a complete list.)

Performance needs to be better, but the server is usable; it uses the X11 portable monchrome framebuffer code. Look in *contrib/server/speedups/donohue.shar* for some ideas which could speed up this server.

SEE ALSO

Xserver(1), xinit(1), X(1)

COPYRIGHT

AUTHORS

Phil Karlton wrote the QVSS-specific pieces of X11; thanks to David Carver of the DEC Workstations group for work on X10.

NAME

Xsun, XsunMono – Sun server for X Version 11

SYNOPSIS

Xsun [option] ...

DESCRIPTION

Xsun is the server for Version 11 of the X window system on Sun hardware. It will normally be started by the *xdm(1)* daemon or by a script that runs the program *xinit(1)*.

CONFIGURATIONS

Xsun supports both color and monochrome screens. *XsunMono* is configured to support only monochrome screens, for low-end systems with limited memory.

If a specific framebuffer device file isn't specified on the command line with the *–dev* switch, the server will look for the following framebuffers (the *constype(1)* program may be used to tell which type of display is attached as the console):

/dev/bwtwo0
This monochrome display is available on Sun-3/50, Sun-3/75, Sun-3/140, Sun-3/260, and Sun-3/280 platforms with both the 1152x900 and 1600x1280 screen sizes.

/dev/cgtwo0
This color display is available on 3/160C, 3/260C, and 4/260C platforms. It may also be used as a monochrome-only display with the *-mono* switch.

/dev/cgthree0
This color display is available on both the Sun386i and SPARCstation 1 platforms.

/dev/cgfour0
This display is available on Sun-3/60, Sun-3/110, and Sun-4/110 platforms. It provides both color and monochrome screens (numbered 0 and 1, respectively by default) on the same monitor by toggling between them whenever the mouse goes off the left or right edges of the screen.

/dev/cgsix0
This color display is available on SPARCstation 1 platforms with the GX graphics accelerator (which is used by the server to make some operations faster).

Finally, if no specific framebuffers are found, the generic framebuffer interface */dev/fb* is used.

Note that the server does not support the GP, GP+, or GP2 graphics coprocessors. On these platforms, the **cgtwo** framebuffer should be used instead (either by creating it or specifying *-dev /dev/cgtwo0* on the server command line).

OPTIONS

In addition to the normal server options described in the *Xserver(1)* manual page, *Xsun* accepts the following command line switches:

–ar1 *milliseconds*
This option specifies amount of time in milliseconds before which a pressed key should begin to autorepeat.

–ar2 *milliseconds*
This option specifies the internal in milliseconds between autorepeats of pressed keys.

–mono When used with the **cgtwo**, this option indicates that the server should emulate a monochrome framebuffer instead of the normal color framebuffer. When used with the **cgfour**, this option indicates that the monochrome screen should be numbered 0 and the color screen numbered 1 (instead of the other way around).

–zaphod This option disables switching between screens by sliding the mouse off the left or right edges. With this disabled, a window manager function must be used to switch between screens.

–debug This option indicates that the server is being run from a debugger, and that it should **not** put its standard input, output and error files into non-blocking mode.

–dev *filename*

This option specifies the name of the framebuffer device file to be used instead of the built-in defaults described above.

USE WITH SUNWINDOWS OR NeWS

The server can be run from outside **suntools**, in which case it configures for all available displays (unless overridden by **-dev** options). Otherwise, it can be run ‘‘on top of’’ either **suntools** or **NeWS**. In this case, it takes over the entire screen it was invoked from.

If you have multiple displays, you can run **suntools** on both, use **adjacentscreens** to move the mouse between them, and then run *Xsun* on top of one of the desktops. **Adjacentscreens** will still be in effect, so you can move between window systems by sliding the mouse from one screen to another.

ENVIRONMENT

XDEVICE

If present, and if no explicit **-dev** options are given, specifies the (colon separated) list of display devices to use.

WINDOW_PARENT

If present, specifies the */dev/win* file of the **suntools** desktop.

SEE ALSO

X(1), Xserver(1), xdm(1), xinit(1)

BUGS

The auto-configuration depends on there being appropriate special files in the */dev* directory for the framebuffers which are to be used. Extra entries can confuse the server. For example, the X/160C in fact has the hardware for a monochrome **bwtwo0** on the CPU board. So if */dev* has a special file for */dev/bwtwo0*, the server will use it, even though there is no monitor attached to the monochrome framebuffer. The server will appear to start, but not to paint a cursor, because the cursor is on the monochrome frame buffer. The solution is to remove the */dev* entries for any device you don’t have a monitor for.

There is a bug in pre-FCS operating systems for the Sun-4 which causes the server to crash driving a **cgtwo**.

There is a race condition that sometimes happens when running ‘‘on top of’’ NeWS. The symptom is that parts of NeWS show through, and that the keyboard is in a non-useful state. There is no simple workaround.

Colon separated devices names do not work.

AUTHORS

U. C. Berkeley

Adam de Boor.

Sun Microsystems

David Rosenthal, Stuart Marks, Robin Schaufler, Mike Schwartz, Frances Ho, Geoff Lee, and Mark Opperman.

MIT Laboratory for Computer Science

Bob Scheifler, Keith Packard

NAME

constype - print type of Sun console

SYNOPSIS

constype

DESCRIPTION

Consoletype prints on the standard output the Sun code for the type of display that the console is. It is one of:

bw?	Black and White, where ? is 1-4. (eg) 3-50s are bw2
cg?	Colour Graphics display, where ? is 1-4
gp?	Optional Graphics Processor board, where ? is 1-2
ns?	Not Sun display - where ? is A-J

This is useful in determining startup values and defaults for window systems.

BUGS

Not tested on all monitor types

COPYRIGHT

AUTHOR

Doug Moran <moran@ai.sri.com>

NAME

kbd_mode – recover the Sun console keyboard

SYNOPSIS

kbd_mode [-a -e -n -u]

DESCRIPTION

Kbd_mode resets the Sun console keyboard to a rational state.

OPTIONS

The following options are supported, see *kb(4S)* for details:

–a Causes ASCII to be reported.

–e Causes *Firm_events* to be reported.

–n Causes up/down key codes to be reported.

–u Causes undecoded keyboard values to be reported.

SEE ALSO

kb(4S)