NAME

XFree86 - X11R6 X server

SYNOPSIS

XFree86 [**:***display*] [*option ...*]

DESCRIPTION

XFree86 is a full featured X server that was originally designed for UNIX and UNIX-like operating systems running on Intel x86 hardware. It now runs on a wider range of hardware and OS platforms.

This work was originally derived from *X386 1.2* by Thomas Roell which was contributed to X11R5 by Snitily Graphics Consulting Service. The **XFree86** server architecture was redesigned for the 4.0 release, and it includes among many other things a loadable module system derived from code donated by Metro Link, Inc. The current XFree86 release is compatible with X11R6.6.

PLATFORMS

XFree86 operates under a wide range of operating systems and hardware platforms. The Intel x86 (IA32) architecture is the most widely supported hardware platform. Other hardware platforms include Compaq Alpha, Intel IA64, SPARC and PowerPC. The most widely supported operating systems are the free/OpenSource UNIX-like systems such as Linux, FreeBSD, NetBSD and OpenBSD. Commercial UNIX operating systems such as Solaris (x86) and UnixWare are also supported. Other supported operating systems include LynxOS, and GNU Hurd. Darwin and Mac OS X are supported with the XDarwin(1) X server. Win32/Cygwin is supported with the XWin X server.

NETWORK CONNECTIONS

XFree86 supports connections made using the following reliable byte-streams:

Local

On most platforms, the "Local" connection type is a UNIX-domain socket. On some System V platforms, the "local" connection types also include STREAMS pipes, named pipes, and some other mechanisms.

TCPIP

XFree86 listens on port 6000+*n*, where *n* is the display number. This connection type can be disabled with the **−nolisten** option (see the Xserver(1) man page for details).

ENVIRONMENT VARIABLES

For operating systems that support local connections other than Unix Domain sockets (SVR3 and SVR4), there is a compiled-in list specifying the order in which local connections should be attempted. This list can be overridden by the *XLOCAL* environment variable described below. If the display name indicates a best-choice connection should be made (e.g. **:0.0**), each connection mechanism is tried until a connection succeeds or no more mechanisms are available. Note: for these OSs, the Unix Domain socket connection is treated differently from the other local connection types. To use it the connection must be made to **unix:0.0**.

The *XLOCAL* environment variable should contain a list of one more more of the following:

NAMED
PTS
SCO
ISC

which represent SVR4 Named Streams pipe, Old-style USL Streams pipe, SCO XSight Streams pipe, and ISC Streams pipe, respectively. You can select a single mechanism (e.g. *XLOCAL=NAMED*), or an ordered list (e.g. *XLOCAL="NAMED:PTS:SCO"*). his variable overrides the compiled-in defaults. For SVR4 it is recommended that *NAMED* be the first preference connection. The default setting is *PTS:NAMED:ISC:SCO*.

To globally override the compiled-in defaults, you should define (and export if using **sh** or **ksh**) *XLOCAL* globally. If you use startx(1) or xinit(1), the definition should be at the top of your *.xinitrc* file. If you use xdm(1), the definitions should be early on in the */usr/X11R6/lib/X11/xdm/Xsession* script.

OPTIONS

XFree86 supports several mechanisms for supplying/obtaining configuration and run-time parameters: command line options, environment variables, the XF86Config(5) configuration file, auto-detection, and fallback defaults. When the same information is supplied in more than one way, the highest precedence mechanism is used. The list of mechanisms is ordered from highest precedence to lowest. Note that not all parameters can be supplied via all methods. The available command line options and environment variables (and some defaults) are described here and in the Xserver(1) manual page. Most configuration file parameters, with their defaults, are described in the XF86Config(5) manual page. Driver and module specific configuration parameters are described in the relevant driver or module manual page.

Starting with version 4.4, **XFree86** has support for generating a usable configuration at run-time when no XF86Config(5) configuration file is provided. The initial version of this automatic configuration support is targeted at the most popular hardware and software platforms supported by XFree86. Some details about how this works can be found in the **CONFIGURATION** section below and in the getconfig(1) manual page.

In addition to the normal server options described in the Xserver(1) manual page, **XFree86** accepts the following command line switches:

vt*XX*
: *XX* specifies the Virtual Terminal device number which **XFree86** will use. Without this option, **XFree86** will pick the first available Virtual Terminal that it can locate. This option applies only to platforms such as Linux, BSD, SVR3 and SVR4, that have virtual terminal support.

−allowMouseOpenFail
: Allow the server to start up even if the mouse device can't be opened or initialised. This is equivalent to the **AllowMouseOpenFail** XF86Config(5) file option.

−allowNonLocalModInDev
: Allow changes to keyboard and mouse settings from non-local clients. By default, connections from non-local clients are not allowed to do this. This is equivalent to the **AllowNonLocalModInDev** XF86Config(5) file option.

−allowNonLocalXvidtune
: Make the VidMode extension available to remote clients. This allows the xvidtune client to connect from another host. This is equivalent to the **AllowNonLocalXvidtune** XF86Config(5) file option. By default non-local connections are not allowed.

−bgamma *value*
: Set the blue gamma correction. *value* must be between 0.1 and 10. The default is 1.0. Not all drivers support this. See also the **−gamma**, **−rgamma**, and **−ggamma** options.

−bpp *n*
: No longer supported. Use **−depth** to set the color depth, and use **−fbbpp** if you really need to force a non-default framebuffer (hardware) pixel format.

−configure
: When this option is specified, the **XFree86** server loads all video driver modules, probes for available hardware, and writes out an initial XF86Config(5) file based on what was detected. This option currently has some problems on some platforms, but in most cases it is a good way to bootstrap the configuration process. This option is only available when the server is run as root (i.e, with real-uid 0).

−crt /dev/tty*XX*
: SCO only. This is the same as the **vt** option, and is provided for compatibility with the native SCO X server.

−depth *n*
: Sets the default color depth. Legal values are 1, 4, 8, 15, 16, and 24. Not all drivers support all values.

−disableModInDev
: Disable dynamic modification of input device settings. This is equivalent to the **DisableModInDev** XF86Config(5) file option.

−disableVidMode
Disable the the parts of the VidMode extension (used by the xvidtune client) that can be used to change the video modes. This is equivalent to the **DisableVidModeExtension** XF86Config(5) file option.

−fbbpp *n*
Sets the number of framebuffer bits per pixel. You should only set this if you're sure it's necessary; normally the server can deduce the correct value from **−depth** above. Useful if you want to run a depth 24 configuration with a 24 bpp framebuffer rather than the (possibly default) 32 bpp framebuffer (or vice versa). Legal values are 1, 8, 16, 24, 32. Not all drivers support all values.

−flipPixels
Swap the default values for the black and white pixels.

−gamma *value*
Set the gamma correction. *value* must be between 0.1 and 10. The default is 1.0. This value is applied equally to the R, G and B values. Those values can be set independently with the **−rgamma**, **−bgamma**, and **−ggamma** options. Not all drivers support this.

−ggamma *value*
Set the green gamma correction. *value* must be between 0.1 and 10. The default is 1.0. Not all drivers support this. See also the **−gamma**, **−rgamma**, and **−bgamma** options.

−ignoreABI
The **XFree86** server checks the ABI revision levels of each module that it loads. It will normally refuse to load modules with ABI revisions that are newer than the server's. This is because such modules might use interfaces that the server does not have. When this option is specified, mismatches like this are downgraded from fatal errors to warnings. This option should be used with care.

−keeptty
Prevent the server from detaching its initial controlling terminal. This option is only useful when debugging the server. Not all platforms support (or can use) this option.

−keyboard *keyboard-name*
Use the XF86Config(5) file **InputDevice** section called *keyboard-name* as the core keyboard. This option is ignored when the **Layout** section specifies a core keyboard. In the absence of both a Layout section and this option, the first relevant **InputDevice** section is used for the core keyboard.

−layout *layout-name*
Use the XF86Config(5) file **Layout** section called *layout-name*. By default the first **Layout** section is used.

−logfile *filename*
Use the file called *filename* as the **XFree86** server log file. The default log file is **/var/log/XFree86.*n*.log** on most platforms, where *n* is the display number of the **XFree86** server. The default may be in a different directory on some platforms. This option is only available when the server is run as root (i.e, with real-uid 0).

−logverbose [*n*]
Sets the verbosity level for information printed to the **XFree86** server log file. If the *n* value isn't supplied, each occurrence of this option increments the log file verbosity level. When the *n* value is supplied, the log file verbosity level is set to that value. The default log file verbosity level is 3.

−modulepath *searchpath*
Set the module search path to *searchpath*. *searchpath* is a comma separated list of directories to search for **XFree86** server modules. This option is only available when the server is run as root (i.e, with real-uid 0).

−nosilk Disable Silken Mouse support.

−pixmap24
Set the internal pixmap format for depth 24 pixmaps to 24 bits per pixel. The default is usually 32 bits per pixel. There is normally little reason to use this option. Some client applications don't like this pixmap format, even though it is a perfectly legal format. This is equivalent to the **Pixmap** XF86Config(5) file option.

−pixmap32
Set the internal pixmap format for depth 24 pixmaps to 32 bits per pixel. This is usually the default. This is equivalent to the **Pixmap** XF86Config(5) file option.

−pointer *pointer-name*
Use the XF86Config(5) file **InputDevice** section called *pointer-name* as the core pointer. This option is ignored when the **Layout** section specifies a core pointer. In the absence of both a Layout section and this option, the first relevant **InputDevice** section is used for the core pointer.

−probeonly
Causes the server to exit after the device probing stage. The XF86Config(5) file is still used when this option is given, so information that can be auto-detected should be commented out.

−quiet Suppress most informational messages at startup. The verbosity level is set to zero.

−rgamma *value*
Set the red gamma correction. *value* must be between 0.1 and 10. The default is 1.0. Not all drivers support this. See also the **−gamma**, **−bgamma**, and **−ggamma** options.

−scanpci
When this option is specified, the **XFree86** server scans the PCI bus, and prints out some information about each device that was detected. See also scanpci(1) and pcitweak(1).

−screen *screen-name*
Use the XF86Config(5) file **Screen** section called *screen-name*. By default the screens referenced by the default **Layout** section are used, or the first **Screen** section when there are no **Layout** sections.

−showconfig
This is the same as the **−version** option, and is included for compatibility reasons. It may be removed in a future release, so the **−version** option should be used instead.

−weight *nnn*
Set RGB weighting at 16 bpp. The default is 565. This applies only to those drivers which support 16 bpp.

−verbose [*n*]
Sets the verbosity level for information printed on stderr. If the *n* value isn't supplied, each occurrence of this option increments the verbosity level. When the *n* value is supplied, the verbosity level is set to that value. The default verbosity level is 0.

−version
Print out the server version, patchlevel, release date, the operating system/platform it was built on, and whether it includes module loader support.

−xf86config *file*
Read the server configuration from *file*. This option will work for any file when the server is run as root (i.e, with real-uid 0), or for files relative to a directory in the config search path for all other users.

KEYBOARD

The **XFree86** server is normally configured to recognize various special combinations of key presses that instruct the server to perform some action, rather than just sending the key press event to a client application. The default XKEYBOARD keymap defines the key combinations listed below. The server also has these key combinations builtin to its event handler for cases where the XKEYBOARD extension is not being used. When using the XKEYBOARD extension, which key combinations perform which actions is completely configurable.

For more information about when the builtin event handler is used to recognize the special key combinations, see the documentation on the **HandleSpecialKeys** option in the XF86Config(5) man page.

The special combinations of key presses recognized directly by **XFree86** are:

Ctrl+Alt+Backspace
Immediately kills the server -- no questions asked. This can be disabled with the **DontZap** XF86Config(5) file option.

Ctrl+Alt+Keypad-Plus
Change video mode to next one specified in the configuration file. This can be disabled with the **DontZoom** XF86Config(5) file option.

Ctrl+Alt+Keypad-Minus
Change video mode to previous one specified in the configuration file. This can be disabled with the **DontZoom** XF86Config(5) file option.

Ctrl+Alt+Keypad-Multiply
Not treated specially by default. If the **AllowClosedownGrabs** XF86Config(5) file option is specified, this key sequence kills clients with an active keyboard or mouse grab as well as killing any application that may have locked the server, normally using the XGrabServer(3) Xlib function.

Ctrl+Alt+Keypad-Divide
Not treated specially by default. If the **AllowDeactivateGrabs** XF86Config(5) file option is specified, this key sequence deactivates any active keyboard and mouse grabs.

Ctrl+Alt+F1...F12
For BSD and Linux systems with virtual terminal support, these keystroke combinations are used to switch to virtual terminals 1 through 12, respectively. This can be disabled with the **DontVTSwitch** XF86Config(5) file option.

CONFIGURATION

XFree86 typically uses a configuration file called **XF86Config** for its initial setup. Refer to the XF86Config(5) manual page for information about the format of this file.

Starting with version 4.4, **XFree86** has a mechanism for automatically generating a built-in configuration at run-time when no **XF86Config** file is present. The current version of this automatic configuration mechanism works in three ways.

The first is via enhancements that have made many components of the **XF86Config** file optional. This means that information that can be probed or reasonably deduced doesn't need to be specified explicitly, greatly reducing the amount of built-in configuration information that needs to be generated at run-time.

The second is to use an external utility called getconfig(1), when available, to use meta-configuration information to generate a suitable configuration for the primary video device. The meta-configuration information can be updated to allow an existing installation to get the best out of new hardware or to work around bugs that are found post-release.

The third is to have "safe" fallbacks for most configuration information. This maximises the likelihood that the **XFree86** server will start up in some usable configuration even when information about the specific hardware is not available.

The automatic configuration support for XFree86 is work in progress. It is currently aimed at the most popular hardware and software platforms supported by XFree86. Enhancements are planned for future releases.

FILES

The **XFree86** server config file can be found in a range of locations. These are documented fully in the XF86Config(5) manual page. The most commonly used locations are shown here.

/etc/X11/XF86Config	Server configuration file.
/etc/X11/XF86Config-4	Server configuration file.
/etc/XF86Config	Server configuration file.
/usr/X11R6/etc/XF86Config	Server configuration file.
/usr/X11R6/lib/X11/XF86Config	Server configuration file.
/var/log/XFree86.*n*.log	Server log file for display *n*.
/usr/X11R6/bin/*	Client binaries.
/usr/X11R6/include/*	Header files.
/usr/X11R6/lib/*	Libraries.
/usr/X11R6/lib/X11/fonts/*	Fonts.
/usr/X11R6/lib/X11/rgb.txt	Color names to RGB mapping.
/usr/X11R6/lib/X11/XErrorDB	Client error message database.
/usr/X11R6/lib/X11/app-defaults/*	Client resource specifications.
/usr/X11R6/man/man?/*	Manual pages.
/etc/X*n*.hosts	Initial access control list for display *n*.

SEE ALSO

X(7), Xserver(1), xdm(1), xinit(1), XF86Config(5), xf86config(1), xf86cfg(1), xvidtune(1), apm(4), ati(4), chips(4), cirrus(4), cyrix(4), fbdev(4), glide(4), glint(4), i128(4), i740(4), i810(4), imstt(4), mga(4), neomagic(4), nsc(4), nv(4), r128(4), rendition(4), s3virge(4), siliconmotion(4), sis(4), sunbw2(4), suncg14(4), suncg3(4), suncg6(4), sunffb(4), sunleo(4), suntcx(4), tdfx(4), tga(4), trident(4), tseng(4), v4l(4), vesa(4), vga(4), vmware(4),
README *<http://www.xfree86.org/current/README.html>*,
RELNOTES *<http://www.xfree86.org/current/RELNOTES.html>*,
README.mouse *<http://www.xfree86.org/current/mouse.html>*,
README.DRI *<http://www.xfree86.org/current/DRI.html>*,
Status *<http://www.xfree86.org/current/Status.html>*,
Install *<http://www.xfree86.org/current/Install.html>*.

AUTHORS

XFree86 has many contributors world wide. The names of most of them can be found in the documentation, CHANGELOG files in the source tree, and in the actual source code.

XFree86 was originally based on *X386 1.2* by Thomas Roell, which was contributed to the then X Consortium's X11R5 distribution by SGCS.

The project that became XFree86 was originally founded in 1992 by David Dawes, Glenn Lai, Jim Tsillas and David Wexelblat.

XFree86 was later integrated in the then X Consortium's X11R6 release by a group of dedicated XFree86 developers, including the following:

Stuart Anderson *anderson@metrolink.com*
Doug Anson *danson@lgc.com*
Gertjan Akkerman *akkerman@dutiba.twi.tudelft.nl*
Mike Bernson *mike@mbsun.mlb.org*
Robin Cutshaw *robin@XFree86.org*
David Dawes *dawes@XFree86.org*
Marc Evans *marc@XFree86.org*
Pascal Haible *haible@izfm.uni-stuttgart.de*

Matthieu Herrb *Matthieu.Herrb@laas.fr*
Dirk Hohndel *hohndel@XFree86.org*
David Holland *davidh@use.com*
Alan Hourihane *alanh@fairlite.demon.co.uk*
Jeffrey Hsu *hsu@soda.berkeley.edu*
Glenn Lai *glenn@cs.utexas.edu*
Ted Lemon *mellon@ncd.com*
Rich Murphey *rich@XFree86.org*
Hans Nasten *nasten@everyware.se*
Mark Snitily *mark@sgcs.com*
Randy Terbush *randyt@cse.unl.edu*
Jon Tombs *tombs@XFree86.org*
Kees Verstoep *versto@cs.vu.nl*
Paul Vixie *paul@vix.com*
Mark Weaver *Mark_Weaver@brown.edu*
David Wexelblat *dwex@XFree86.org*
Philip Wheatley *Philip.Wheatley@ColumbiaSC.NCR.COM*
Thomas Wolfram *wolf@prz.tu-berlin.de*
Orest Zborowski *orestz@eskimo.com*

The current XFree86 core team consists of:

David Dawes *dawes@xfree86.org*
Egbert Eich *eich@xfree86.org*
Marc Evans *marc@xfree86.org*
Matthieu Herrb *herrb@xfree86.org*
Alan Hourihane *alanh@xfree86.org*
Marc La France *tsi@xfree86.org*
Kevin Martin *martin@xfree86.org*
Rich Murphey *rich@xfree86.org*
Mark Vojkovich *markv@xfree86.org*
David Wexelblat *dwex@xfree86.org*

XFree86 source is available from the FTP server <*ftp://ftp.XFree86.org/pub/XFree86/*>, and from the XFree86 CVS server <*http://www.xfree86.org/cvs/*>. Documentation and other information can be found from the XFree86 web site <*http://www.xfree86.org/*>.

LEGAL

NAME

XF86Config - Configuration File for XFree86

INTRODUCTION

XFree86 supports several mechanisms for supplying/obtaining configuration and run-time parameters: command line options, environment variables, the XF86Config configuration file, auto-detection, and fallback defaults. When the same information is supplied in more than one way, the highest precedence mechanism is used. The list of mechanisms is ordered from highest precedence to lowest. Note that not all parameters can be supplied via all methods. The available command line options and environment variables (and some defaults) are described in the Xserver(1) and XFree86(1) manual pages. Most configuration file parameters, with their defaults, are described below. Driver and module specific configuration parameters are described in the relevant driver or module manual page.

Starting with version 4.4, **XFree86** has support for generating a usable configuration at run-time when no **XF86Config** file is provided. The initial version of this automatic configuration support is targeted at the most popular hardware and software platforms supported by XFree86. Some details about how this works can be found in the XFree86(1) and getconfig(1) manual pages.

DESCRIPTION

XFree86 uses a configuration file called **XF86Config** for its initial setup. This configuration file is searched for in the following places when the server is started as a normal user:

/etc/X11/<cmdline>
/usr/X11R6/etc/X11/<cmdline>
*/etc/X11/***$XF86CONFIG**
*/usr/X11R6/etc/X11/***$XF86CONFIG**
/etc/X11/XF86Config-4
/etc/X11/XF86Config
/etc/XF86Config
/usr/X11R6/etc/X11/XF86Config.<hostname>
/usr/X11R6/etc/X11/XF86Config-4
/usr/X11R6/etc/X11/XF86Config
/usr/X11R6/lib/X11/XF86Config.<hostname>
/usr/X11R6/lib/X11/XF86Config-4
/usr/X11R6/lib/X11/XF86Config

where *<cmdline>* is a relative path (with no ".." components) specified with the **−xf86config** command line option, **$XF86CONFIG** is the relative path (with no ".." components) specified by that environment variable, and *<hostname>* is the machine's hostname as reported by gethostname(3).

When the XFree86 server is started by the "root" user, the config file search locations are as follows:

<cmdline>
/etc/X11/<cmdline>
/usr/X11R6/etc/X11/<cmdline>
$XF86CONFIG
*/etc/X11/***$XF86CONFIG**
*/usr/X11R6/etc/X11/***$XF86CONFIG**
$HOME*/XF86Config*
/etc/X11/XF86Config-4
/etc/X11/XF86Config
/etc/XF86Config
/usr/X11R6/etc/X11/XF86Config.<hostname>
/usr/X11R6/etc/X11/XF86Config-4
/usr/X11R6/etc/X11/XF86Config
/usr/X11R6/lib/X11/XF86Config.<hostname>
/usr/X11R6/lib/X11/XF86Config-4
/usr/X11R6/lib/X11/XF86Config

where <*cmdline*> is the path specified with the **−xf86config** command line option (which may be absolute or relative), **$XF86CONFIG** is the path specified by that environment variable (absolute or relative), **$HOME** is the path specified by that environment variable (usually the home directory), and <*hostname*> is the machine's hostname as reported by gethostname(3).

The **XF86Config** file is composed of a number of sections which may be present in any order. Each section has the form:

Section "*SectionName***"**
SectionEntry
...
EndSection

The section names are:

Files File pathnames
ServerFlags Server flags
Module Dynamic module loading
InputDevice Input device description
Device Graphics device description
VideoAdaptor Xv video adaptor description
Monitor Monitor description
Modes Video modes descriptions
Screen Screen configuration
ServerLayout Overall layout
DRI DRI-specific configuration
Vendor Vendor-specific configuration

The following obsolete section names are still recognised for compatibility purposes. In new config files, the **InputDevice** section should be used instead.

Keyboard Keyboard configuration
Pointer Pointer/mouse configuration

The old **XInput** section is no longer recognised.

The **ServerLayout** sections are at the highest level. They bind together the input and output devices that will be used in a session. The input devices are described in the **InputDevice** sections. Output devices usually consist of multiple independent components (e.g., and graphics board and a monitor). These multiple components are bound together in the **Screen** sections, and it is these that are referenced by the **ServerLayout** section. Each **Screen** section binds together a graphics board and a monitor. The graphics boards are described in the **Device** sections, and the monitors are described in the **Monitor** sections.

Config file keywords are case-insensitive, and "_" characters are ignored. Most strings (including **Option** names) are also case-insensitive, and insensitive to white space and "_" characters.

Each config file entry usually takes up a single line in the file. They consist of a keyword, which is possibly followed by one or more arguments, with the number and types of the arguments depending on the keyword. The argument types are:

Integer an integer number in decimal, hex or octal
Real a floating point number
String a string enclosed in double quote marks (")

Note: hex integer values must be prefixed with "0x", and octal values with "0".

A special keyword called **Option** may be used to provide free-form data to various components of the server. The **Option** keyword takes either one or two string arguments. The first is the option name, and the optional second argument is the option value. Some commonly used option value types include:

Integer an integer number in decimal, hex or octal
Real a floating point number
String a sequence of characters

Boolean a boolean value (see below)
Frequency a frequency value (see below)

Note that *all* **Option** values, not just strings, must be enclosed in quotes.

Boolean options may optionally have a value specified. When no value is specified, the option's value is **TRUE**. The following boolean option values are recognised as **TRUE**:

1, **on**, **true**, **yes**

and the following boolean option values are recognised as **FALSE**:

0, **off**, **false**, **no**

If an option name is prefixed with "**No**", then the option value is negated.

Example: the following option entries are equivalent:

Option "Accel" "Off"
Option "NoAccel"
Option "NoAccel" "On"
Option "Accel" "false"
Option "Accel" "no"

Frequency option values consist of a real number that is optionally followed by one of the following frequency units:

Hz, **k**, **kHz**, **M**, **MHz**

When the unit name is omitted, the correct units will be determined from the value and the expectations of the appropriate range of the value. It is recommended that the units always be specified when using frequency option values to avoid any errors in determining the value.

FILES SECTION

The **Files** section is used to specify some path names required by the server. Some of these paths can also be set from the command line (see Xserver(1) and XFree86(1)). The command line settings override the values specified in the config file. The **Files** section is optional, as are all of the entries that may appear in it.

The entries that can appear in this section are:

FontPath "*path***"**

sets the search path for fonts. This path is a comma separated list of font path elements which the XFree86 server searches for font databases. Multiple **FontPath** entries may be specified, and they will be concatenated to build up the fontpath used by the server. Font path elements may be either absolute directory paths, or a font server identifier. Font server identifiers have the form:

<trans>/*<hostname>*:*<port-number>*

where *<trans>* is the transport type to use to connect to the font server (e.g., **unix** for UNIX-domain sockets or **tcp** for a TCP/IP connection), *<hostname>* is the hostname of the machine running the font server, and *<port-number>* is the port number that the font server is listening on (usually 7100).

When this entry is not specified in the config file, the server falls back to the compiled-in default font path, which contains the following font path elements:

/usr/X11R6/lib/X11/fonts/misc/
/usr/X11R6/lib/X11/fonts/Speedo/
/usr/X11R6/lib/X11/fonts/Type1/
/usr/X11R6/lib/X11/fonts/CID/
/usr/X11R6/lib/X11/fonts/75dpi/
/usr/X11R6/lib/X11/fonts/100dpi/

The recommended font path contains the following font path elements:

/usr/X11R6/lib/X11/fonts/local/
/usr/X11R6/lib/X11/fonts/misc/
/usr/X11R6/lib/X11/fonts/75dpi/:unscaled
/usr/X11R6/lib/X11/fonts/100dpi/:unscaled
/usr/X11R6/lib/X11/fonts/Type1/
/usr/X11R6/lib/X11/fonts/CID/
/usr/X11R6/lib/X11/fonts/Speedo/
/usr/X11R6/lib/X11/fonts/75dpi/
/usr/X11R6/lib/X11/fonts/100dpi/

Font path elements that are found to be invalid are removed from the font path when the server starts up.

RGBPath "*path***"**
sets the path name for the RGB color database. When this entry is not specified in the config file, the server falls back to the compiled-in default RGB path, which is:

/usr/X11R6/lib/X11/rgb

Note that an implicit *.txt* is added to this path if the server was compiled to use text rather than binary format RGB color databases.

ModulePath "*path***"**
sets the search path for loadable XFree86 server modules. This path is a comma separated list of directories which the XFree86 server searches for loadable modules loading in the order specified. Multiple **ModulePath** entries may be specified, and they will be concatenated to build the module search path used by the server.

SERVERFLAGS SECTION

The **ServerFlags** section is used to specify some global XFree86 server options. All of the entries in this section are **Options**, although for compatibility purposes some of the old style entries are still recognised. Those old style entries are not documented here, and using them is discouraged. The **ServerFlags** section is optional, as are the entries that may be specified in it.

Options specified in this section (with the exception of the **"DefaultServerLayout" Option**) may be overridden by **Options** specified in the active **ServerLayout** section. Options with command line equivalents are overridden when their command line equivalent is used. The options recognised by this section are:

Option "DefaultServerLayout" "*layout-id***"**
This specifies the default **ServerLayout** section to use in the absence of the **−layout** command line option.

Option "NoTrapSignals" "*boolean***"**
This prevents the XFree86 server from trapping a range of unexpected fatal signals and exiting cleanly. Instead, the XFree86 server will die and drop core where the fault occurred. The default behaviour is for the XFree86 server to exit cleanly, but still drop a core file. In general you never want to use this option unless you are debugging an XFree86 server problem and know how to deal with the consequences.

Option "DontVTSwitch" "*boolean***"**
This disallows the use of the **Ctrl+Alt+F***n* sequence (where F*n* refers to one of the numbered function keys). That sequence is normally used to switch to another "virtual terminal" on operating systems that have this feature. When this option is enabled, that key sequence has no special meaning and is passed to clients. Default: off.

Option "DontZap" "*boolean***"**
This disallows the use of the **Ctrl+Alt+Backspace** sequence. That sequence is normally used to terminate the XFree86 server. When this option is enabled, that key sequence has no special meaning and is passed to clients. Default: off.

Option "DontZoom" "*boolean*"
This disallows the use of the **Ctrl+Alt+Keypad-Plus** and **Ctrl+Alt+Keypad-Minus** sequences. These sequences allows you to switch between video modes. When this option is enabled, those key sequences have no special meaning and are passed to clients. Default: off.

Option "DisableVidModeExtension" "*boolean*"
This disables the parts of the VidMode extension used by the xvidtune client that can be used to change the video modes. Default: the VidMode extension is enabled.

Option "AllowNonLocalXvidtune" "*boolean*"
This allows the xvidtune client (and other clients that use the VidMode extension) to connect from another host. Default: off.

Option "DisableModInDev" "*boolean*"
This disables the parts of the XFree86-Misc extension that can be used to modify the input device settings dynamically. Default: that functionality is enabled.

Option "AllowNonLocalModInDev" "*boolean*"
This allows a client to connect from another host and change keyboard and mouse settings in the running server. Default: off.

Option "AllowMouseOpenFail" "*boolean*"
This allows the server to start up even if the mouse device can't be opened/initialised. Default: false.

Option "VTInit" "*command*"
Runs *command* after the VT used by the server has been opened. The command string is passed to "/bin/sh -c", and is run with the real user's id with stdin and stdout set to the VT. The purpose of this option is to allow system dependent VT initialisation commands to be run. This option should rarely be needed. Default: not set.

Option "VTSysReq" "*boolean*"
enables the SYSV-style VT switch sequence for non-SYSV systems which support VT switching. This sequence is **Alt-SysRq** followed by a function key (**Fn**). This prevents the XFree86 server trapping the keys used for the default VT switch sequence, which means that clients can access them. Default: off.

Option "XkbDisable" "*boolean*"
disable/enable the XKEYBOARD extension. The −kb command line option overrides this config file option. Default: XKB is enabled.

Option "BlankTime" "*time*"
sets the inactivity timeout for the blanking phase of the screensaver. *time* is in minutes. This is equivalent to the XFree86 server's '-s' flag, and the value can be changed at run-time with xset(1). Default: 10 minutes.

Option "StandbyTime" "*time*"
sets the inactivity timeout for the "standby" phase of DPMS mode. *time* is in minutes, and the value can be changed at run-time with xset(1). Default: 20 minutes. This is only suitable for VESA DPMS compatible monitors, and may not be supported by all video drivers. It is only enabled for screens that have the **"DPMS"** option set (see the MONITOR section below).

Option "SuspendTime" "*time*"
sets the inactivity timeout for the "suspend" phase of DPMS mode. *time* is in minutes, and the value can be changed at run-time with xset(1). Default: 30 minutes. This is only suitable for VESA DPMS compatible monitors, and may not be supported by all video drivers. It is only enabled for screens that have the **"DPMS"** option set (see the MONITOR section below).

Option "OffTime" "*time*"
sets the inactivity timeout for the "off" phase of DPMS mode. *time* is in minutes, and the value can be changed at run-time with xset(1). Default: 40 minutes. This is only suitable for VESA DPMS compatible monitors, and may not be supported by all video drivers. It is only enabled for

screens that have the **"DPMS"** option set (see the MONITOR section below).

Option "Pixmap" "*bpp***"**
This sets the pixmap format to use for depth 24. Allowed values for *bpp* are 24 and 32. Default: 32 unless driver constraints don't allow this (which is rare). Note: some clients don't behave well when this value is set to 24.

Option "PC98" "*boolean***"**
Specify that the machine is a Japanese PC-98 machine. This should not be enabled for anything other than the Japanese-specific PC-98 architecture. Default: auto-detected.

Option "NoPM" "*boolean***"**
Disables something to do with power management events. Default: PM enabled on platforms that support it.

Option "Xinerama" "*boolean***"**
enable or disable XINERAMA extension. Default is disabled.

Option "AllowDeactivateGrabs" "*boolean***"**
This option enables the use of the **Ctrl+Alt+Keypad-Divide** key sequence to deactivate any active keyboard and mouse grabs. Default: off.

Option "AllowClosedownGrabs" "*boolean***"**
This option enables the use of the **Ctrl+Alt+Keypad-Multiply** key sequence to kill clients with an active keyboard or mouse grab as well as killing any application that may have locked the server, normally using the XGrabServer(3) Xlib function. Default: off.
Note that the options **AllowDeactivateGrabs** and **AllowClosedownGrabs** will allow users to remove the grab used by screen saver/locker programs. An API was written to such cases. If you enable this option, make sure your screen saver/locker is updated.

Option "HandleSpecialKeys" "*when***"**
This option controls when the server uses the builtin handler to process special key combinations (such as **Ctrl+Alt+Backspace**). Normally the XKEYBOARD extension keymaps will provide mappings for each of the special key combinations, so the builtin handler is not needed unless the XKEYBOARD extension is disabled. The value of *when* can be **Always**, **Never**, or **WhenNeeded**. Default: Use the builtin handler only if needed. The server will scan the keymap for a mapping to the **Terminate** action and, if found, use XKEYBOARD for processing actions, otherwise the builtin handler will be used.

MODULE SECTION

The **Module** section is used to specify which XFree86 server modules should be loaded. This section is ignored when the XFree86 server is built in static form. The types of modules normally loaded in this section are XFree86 server extension modules, and font rasteriser modules. Most other module types are loaded automatically when they are needed via other mechanisms. The **Module** section is optional, as are all of the entries that may be specified in it.

Entries in this section may be in two forms. The first and most commonly used form is an entry that uses the **Load** keyword, as described here:

Load "*modulename***"**
This instructs the server to load the module called *modulename*. The module name given should be the module's standard name, not the module file name. The standard name is case-sensitive, and does not include the "lib" prefix, or the ".a", ".o", or ".so" suffixes.

Example: the Type 1 font rasteriser can be loaded with the following entry:

Load "type1"

The second form of entry is a **SubSection,** with the subsection name being the module name, and the contents of the **SubSection** being **Options** that are passed to the module when it is loaded.

Example: the extmod module (which contains a miscellaneous group of server extensions) can be loaded, with the XFree86-DGA extension disabled by using the following entry:

```
SubSection "extmod"
  Option  "omit XFree86-DGA"
EndSubSection
```

Modules are searched for in each directory specified in the **ModulePath** search path, and in the drivers, input, extensions, fonts, and internal subdirectories of each of those directories. In addition to this, operating system specific subdirectories of all the above are searched first if they exist.

To see what font and extension modules are available, check the contents of the following directories:

```
/usr/X11R6/lib/modules/fonts
/usr/X11R6/lib/modules/extensions
```

The "bitmap" font modules is loaded automatically. It is recommended that at very least the "extmod" extension module be loaded. If it isn't some commonly used server extensions (like the SHAPE extension) will not be available.

INPUTDEVICE SECTION

The config file may have multiple **InputDevice** sections. There will normally be at least two: one for the core (primary) keyboard, and one of the core pointer. If either of these two is missing, a default configuration for the missing ones will be used. Currently the default configuration may not work as expected on all platforms.

InputDevice sections have the following format:

```
Section "InputDevice"
   Identifier "name"
   Driver     "inputdriver"
   options
   ...
EndSection
```

The **Identifier** and **Driver** entries are required in all **InputDevice** sections. All other entries are optional.

The **Identifier** entry specifies the unique name for this input device. The **Driver** entry specifies the name of the driver to use for this input device. When using the loadable server, the input driver module "*inputdriver*" will be loaded for each active **InputDevice** section. An **InputDevice** section is considered active if it is referenced by an active **ServerLayout** section, if it is referenced by the **−keyboard** or **−pointer** command line options, or if it is selected implicitly as the core pointer or keyboard device in the absence of such explicit references. The most commonly used input drivers are "keyboard" and "mouse".

In the absence of an explicitly specified core input device, the first **InputDevice** marked as **CorePointer** (or **CoreKeyboard**) is used. If there is no match there, the first **InputDevice** that uses the "mouse" (or "keyboard" or "kbd") driver is used. The final fallback is to use built-in default configurations.

InputDevice sections recognise some driver-independent **Options**, which are described here. See the individual input driver manual pages for a description of the device-specific options.

Option "CorePointer"
When this is set, the input device is installed as the core (primary) pointer device. There must be exactly one core pointer. If this option is not set here, or in the **ServerLayout** section, or from the **−pointer** command line option, then the first input device that is capable of being used as a core pointer will be selected as the core pointer. This option is implicitly set when the obsolete **Pointer** section is used.

Option "CoreKeyboard"
When this is set, the input device is to be installed as the core (primary) keyboard device. There must be exactly one core keyboard. If this option is not set here, in the **ServerLayout** section, or from the **−keyboard** command line option, then the first input device that is capable of being used as a core keyboard will be selected as the core keyboard. This option is implicitly set when the obsolete **Keyboard** section is used.

Option "AlwaysCore" "*boolean***"**

Option "SendCoreEvents" "*boolean***"**
Both of these options are equivalent, and when enabled cause the input device to always report core events. This can be used, for example, to allow an additional pointer device to generate core pointer events (like moving the cursor, etc).

Option "HistorySize" "*number***"**
Sets the motion history size. Default: 0.

Option "SendDragEvents" "*boolean***"**
???

DEVICE SECTION

The config file may have multiple **Device** sections. There must be at least one, for the video card being used.

Device sections have the following format:

Section "Device"
  Identifier "*name***"**
  Driver "*driver***"**
  entries
  ...
EndSection

The **Identifier** and **Driver** entries are required in all **Device** sections. All other entries are optional.

The **Identifier** entry specifies the unique name for this graphics device. The **Driver** entry specifies the name of the driver to use for this graphics device. When using the loadable server, the driver module "*driver*" will be loaded for each active **Device** section. A **Device** section is considered active if it is referenced by an active **Screen** section.

Device sections recognise some driver-independent entries and **Options**, which are described here. Not all drivers make use of these driver-independent entries, and many of those that do don't require them to be specified because the information is auto-detected. See the individual graphics driver manual pages for further information about this, and for a description of the device-specific options. Note that most of the **Options** listed here (but not the other entries) may be specified in the **Screen** section instead of here in the **Device** section.

BusID "*bus-id***"**
This specifies the bus location of the graphics card. For PCI/AGP cards, the *bus-id* string has the form **PCI:***bus***:***device***:***function* (e.g., "PCI:1:0:0" might be appropriate for an AGP card). This field is usually optional in single-head configurations when using the primary graphics card. In multi-head configurations, or when using a secondary graphics card in a single-head configuration, this entry is mandatory. Its main purpose is to make an unambiguous connection between the device section and the hardware it is representing. This information can usually be found by running the XFree86 server with the **−scanpci** command line option.

Screen *number*
This option is mandatory for cards where a single PCI entity can drive more than one display (i.e., multiple CRTCs sharing a single graphics accelerator and video memory). One **Device** section is required for each head, and this parameter determines which head each of the **Device** sections applies to. The legal values of *number* range from 0 to one less than the total number of heads per entity. Most drivers require that the primary screen (0) be present.

Chipset "*chipset***"**
This usually optional entry specifies the chipset used on the graphics board. In most cases this entry is not required because the drivers will probe the hardware to determine the chipset type. Don't specify it unless the driver-specific documentation recommends that you do.

Ramdac "*ramdac-type*"
This optional entry specifies the type of RAMDAC used on the graphics board. This is only used by a few of the drivers, and in most cases it is not required because the drivers will probe the hardware to determine the RAMDAC type where possible. Don't specify it unless the driver-specific documentation recommends that you do.

DacSpeed *speed*

DacSpeed *speed-8 speed-16 speed-24 speed-32*
This optional entry specifies the RAMDAC speed rating (which is usually printed on the RAMDAC chip). The speed is in MHz. When one value is given, it applies to all framebuffer pixel sizes. When multiple values are give, they apply to the framebuffer pixel sizes 8, 16, 24 and 32 respectively. This is not used by many drivers, and only needs to be specified when the speed rating of the RAMDAC is different from the defaults built in to driver, or when the driver can't auto-detect the correct defaults. Don't specify it unless the driver-specific documentation recommends that you do.

Clocks *clock ...*
specifies the pixel that are on your graphics board. The clocks are in MHz, and may be specified as a floating point number. The value is stored internally to the nearest kHz. The ordering of the clocks is important. It must match the order in which they are selected on the graphics board. Multiple **Clocks** lines may be specified, and each is concatenated to form the list. Most drivers do not use this entry, and it is only required for some older boards with non-programmable clocks. Don't specify this entry unless the driver-specific documentation explicitly recommends that you do.

ClockChip "*clockchip-type*"
This optional entry is used to specify the clock chip type on graphics boards which have a programmable clock generator. Only a few XFree86 drivers support programmable clock chips. For details, see the appropriate driver manual page.

VideoRam *mem*
This optional entry specifies the amount of video ram that is installed on the graphics board. This is measured in kBytes. In most cases this is not required because the XFree86 server probes the graphics board to determine this quantity. The driver-specific documentation should indicate when it might be needed.

BiosBase *baseaddress*
This optional entry specifies the base address of the video BIOS for the VGA board. This address is normally auto-detected, and should only be specified if the driver-specific documentation recommends it.

MemBase *baseaddress*
This optional entry specifies the memory base address of a graphics board's linear frame buffer. This entry is not used by many drivers, and it should only be specified if the driver-specific documentation recommends it.

IOBase *baseaddress*
This optional entry specifies the IO base address. This entry is not used by many drivers, and it should only be specified if the driver-specific documentation recommends it.

ChipID *id*
This optional entry specifies a numerical ID representing the chip type. For PCI cards, it is usually the device ID. This can be used to override the auto-detection, but that should only be done when the driver-specific documentation recommends it.

ChipRev *rev*
This optional entry specifies the chip revision number. This can be used to override the auto-detection, but that should only be done when the driver-specific documentation recommends it.

TextClockFreq *freq*
This optional entry specifies the pixel clock frequency that is used for the regular text mode. The frequency is specified in MHz. This is rarely used.

Options
Option flags may be specified in the **Device** sections. These include driver-specific options and driver-independent options. The former are described in the driver-specific documentation. Some of the latter are described below in the section about the **Screen** section, and they may also be included here.

VIDEOADAPTOR SECTION

Nobody wants to say how this works. Maybe nobody knows ...

MONITOR SECTION

The config file may have multiple **Monitor** sections. There should normally be at least one, for the monitor being used, but a default configuration will be created when one isn't specified.

Monitor sections have the following format:

Section "Monitor"
  Identifier "*name***"**
  entries
  ...
EndSection

The only mandatory entry in a **Monitor** section is the **Identifier** entry.

The **Identifier** entry specifies the unique name for this monitor. The **Monitor** section provides information about the specifications of the monitor, monitor-specific **Options**, and information about the video modes to use with the monitor. Specifying video modes is optional because the server now has a built-in list of VESA standard modes. When modes are specified explicitly in the **Monitor** section (with the **Modes**, **ModeLine**, or **UseModes** keywords), built-in modes with the same names are not included. Built-in modes with different names are, however, still implicitly included.

The entries that may be used in **Monitor** sections are described below.

VendorName "*vendor***"**
This optional entry specifies the monitor's manufacturer.

ModelName "*model***"**
This optional entry specifies the monitor's model.

HorizSync *horizsync-range*
gives the range(s) of horizontal sync frequencies supported by the monitor. *horizsync-range* may be a comma separated list of either discrete values or ranges of values. A range of values is two values separated by a dash. By default the values are in units of kHz. They may be specified in MHz or Hz if **MHz** or **Hz** is added to the end of the line. The data given here is used by the XFree86 server to determine if video modes are within the specifications of the monitor. This information should be available in the monitor's handbook. If this entry is omitted, a default range of 28−33kHz is used.

VertRefresh *vertrefresh-range*
gives the range(s) of vertical refresh frequencies supported by the monitor. *vertrefresh-range* may be a comma separated list of either discrete values or ranges of values. A range of values is two values separated by a dash. By default the values are in units of Hz. They may be specified in MHz or kHz if **MHz** or **kHz** is added to the end of the line. The data given here is used by the XFree86 server to determine if video modes are within the specifications of the monitor. This information should be available in the monitor's handbook. If this entry is omitted, a default range of 43-72Hz is used.

DisplaySize *width height*
This optional entry gives the width and height, in millimetres, of the picture area of the monitor. If given this is used to calculate the horizontal and vertical pitch (DPI) of the screen.

Gamma *gamma-value*

Gamma *red-gamma green-gamma blue-gamma*
This is an optional entry that can be used to specify the gamma correction for the monitor. It may be specified as either a single value or as three separate RGB values. The values should be in the range 0.1 to 10.0, and the default is 1.0. Not all drivers are capable of using this information.

UseModes "*modesection-id***"**
Include the set of modes listed in the **Modes** section called *modesection-id.* This make all of the modes defined in that section available for use by this monitor.

Mode "*name***"**
This is an optional multi-line entry that can be used to provide definitions for video modes for the monitor. In most cases this isn't necessary because the built-in set of VESA standard modes will be sufficient. The **Mode** keyword indicates the start of a multi-line video mode description. The mode description is terminated with the **EndMode** keyword. The mode description consists of the following entries:

DotClock *clock*
is the dot (pixel) clock rate to be used for the mode.

HTimings *hdisp hsyncstart hsyncend htotal*
specifies the horizontal timings for the mode.

VTimings *vdisp vsyncstart vsyncend vtotal*
specifies the vertical timings for the mode.

Flags "*flag***"** *...*
specifies an optional set of mode flags, each of which is a separate string in double quotes. **"Interlace"** indicates that the mode is interlaced. **"DoubleScan"** indicates a mode where each scanline is doubled. **"+HSync"** and **"−HSync"** can be used to select the polarity of the HSync signal. **"+VSync"** and **"−VSync"** can be used to select the polarity of the VSync signal. **"Composite"** can be used to specify composite sync on hardware where this is supported. Additionally, on some hardware, **"+CSync"** and **"−CSync"** may be used to select the composite sync polarity.

HSkew *hskew*
specifies the number of pixels (towards the right edge of the screen) by which the display enable signal is to be skewed. Not all drivers use this information. This option might become necessary to override the default value supplied by the server (if any). "Roving" horizontal lines indicate this value needs to be increased. If the last few pixels on a scan line appear on the left of the screen, this value should be decreased.

VScan *vscan*
specifies the number of times each scanline is painted on the screen. Not all drivers use this information. Values less than 1 are treated as 1, which is the default. Generally, the **"DoubleScan" Flag** mentioned above doubles this value.

ModeLine "*name***"** *mode-description*
This entry is a more compact version of the **Mode** entry, and it also can be used to specify video modes for the monitor. is a single line format for specifying video modes. In most cases this isn't necessary because the built-in set of VESA standard modes will be sufficient.

The *mode-description* is in four sections, the first three of which are mandatory. The first is the dot (pixel) clock. This is a single number specifying the pixel clock rate for the mode in MHz. The second section is a list of four numbers specifying the horizontal timings. These numbers are the *hdisp*, *hsyncstart*, *hsyncend*, and *htotal* values. The third section is a list of four numbers specifying the vertical timings. These numbers are the *vdisp*, *vsyncstart*, *vsyncend*, and *vtotal* values.

The final section is a list of flags specifying other characteristics of the mode. **Interlace** indicates that the mode is interlaced. **DoubleScan** indicates a mode where each scanline is doubled. **+HSync** and **−HSync** can be used to select the polarity of the HSync signal. **+VSync** and **−VSync** can be used to select the polarity of the VSync signal. **Composite** can be used to specify composite sync on hardware where this is supported. Additionally, on some hardware, **+CSync** and **−CSync** may be used to select the composite sync polarity. The **HSkew** and **VScan** options mentioned above in the **Modes** entry description can also be used here.

Options

Some **Option** flags that may be useful to include in **Monitor** sections (when needed) include **"DPMS"**, and **"SyncOnGreen"**.

MODES SECTION

The config file may have multiple **Modes** sections, or none. These sections provide a way of defining sets of video modes independently of the **Monitor** sections. **Monitor** sections may include the definitions provided in these sections by using the **UseModes** keyword. In most cases the **Modes** sections are not necessary because the built-in set of VESA standard modes will be sufficient.

Modes sections have the following format:

```
Section "Modes"
  Identifier "name"
  entries
  ...
EndSection
```

The **Identifier** entry specifies the unique name for this set of mode descriptions. The other entries permitted in **Modes** sections are the **Mode** and **ModeLine** entries that are described above in the **Monitor** section.

SCREEN SECTION

The config file may have multiple **Screen** sections. There must be at least one, for the "screen" being used. A "screen" represents the binding of a graphics device (**Device** section) and a monitor (**Monitor** section). A **Screen** section is considered "active" if it is referenced by an active **ServerLayout** section or by the **−screen** command line option. If neither of those is present, the first **Screen** section found in the config file is considered the active one.

Screen sections have the following format:

```
Section "Screen"
  Identifier "name"
  Device     "devid"
  Monitor    "monid"
  entries
  ...
  SubSection "Display"
    entries
    ...
  EndSubSection
  ...
EndSection
```

The **Identifier** and **Device** entries are mandatory. All others are optional.

The **Identifier** entry specifies the unique name for this screen. The **Screen** section provides information specific to the whole screen, including screen-specific **Options**. In multi-head configurations, there will be multiple active **Screen** sections, one for each head. The entries available for this section are:

Device "*device-id*"
This mandatory entry specifies the **Device** section to be used for this screen. This is what ties a specific graphics card to a screen. The *device-id* must match the **Identifier** of a **Device** section in the config file.

Monitor "*monitor-id*"
specifies which monitor description is to be used for this screen. If a **Monitor** name is not specified, a default configuration is used. Currently the default configuration may not function as expected on all platforms.

VideoAdaptor "*xv-id*"
specifies an optional Xv video adaptor description to be used with this screen.

DefaultDepth *depth*
specifies which color depth the server should use by default. The **−depth** command line option can be used to override this. If neither is specified, the default depth is driver-specific, but in most cases is 8.

DefaultFbBpp *bpp*
specifies which framebuffer layout to use by default. The **−fbbpp** command line option can be used to override this. In most cases the driver will chose the best default value for this. The only case where there is even a choice in this value is for depth 24, where some hardware supports both a packed 24 bit framebuffer layout and a sparse 32 bit framebuffer layout.

Options
Various **Option** flags may be specified in the **Screen** section. Some are driver-specific and are described in the driver documentation. Others are driver-independent, and will eventually be described here.

Option "Accel"
Enables XAA (X Acceleration Architecture), a mechanism that makes video cards' 2D hardware acceleration available to the XFree86 server. This option is on by default, but it may be necessary to turn it off if there are bugs in the driver. There are many options to disable specific accelerated operations, listed below. Note that disabling an operation will have no effect if the operation is not accelerated (whether due to lack of support in the hardware or in the driver).

Option "BiosLocation" "*address*"
Set the location of the BIOS for the Int10 module. One may select a BIOS of another card for posting or the legacy V_BIOS range located at 0xc0000 or an alternative address (BUS_ISA). This is only useful under very special circumstances and should be used with extreme care.

Option "InitPrimary" "*boolean*"
Use the Int10 module to initialize the primary graphics card. Normally, only secondary cards are soft-booted using the Int10 module, as the primary card has already been initialized by the BIOS at boot time. Default: false.

Option "NoInt10" "*boolean*"
Disables the Int10 module, a module that uses the int10 call to the BIOS of the graphics card to initialize it. Default: false.

Option "NoMTRR"
Disables MTRR (Memory Type Range Register) support, a feature of modern processors which can improve video performance by a factor of up to 2.5. Some hardware has buggy MTRR support, and some video drivers have been known to exhibit problems when MTRR's are used.

Option "XaaNoCPUToScreenColorExpandFill"
Disables accelerated rectangular expansion blits from source patterns stored in system memory (using a memory-mapped aperture).

Option "XaaNoColor8x8PatternFillRect"
Disables accelerated fills of a rectangular region with a full-color pattern.

Option "XaaNoColor8x8PatternFillTrap"
Disables accelerated fills of a trapezoidal region with a full-color pattern.

Option "XaaNoDashedBresenhamLine"
Disables accelerated dashed Bresenham line draws.

Option "XaaNoDashedTwoPointLine"
Disables accelerated dashed line draws between two arbitrary points.

Option "XaaNoImageWriteRect"
Disables accelerated transfers of full-color rectangular patterns from system memory to video memory (using a memory-mapped aperture).

Option "XaaNoMono8x8PatternFillRect"
Disables accelerated fills of a rectangular region with a monochrome pattern.

Option "XaaNoMono8x8PatternFillTrap"
Disables accelerated fills of a trapezoidal region with a monochrome pattern.

Option "XaaNoOffscreenPixmaps"
Disables accelerated draws into pixmaps stored in offscreen video memory.

Option "XaaNoPixmapCache"
Disables caching of patterns in offscreen video memory.

Option "XaaNoScanlineCPUToScreenColorExpandFill"
Disables accelerated rectangular expansion blits from source patterns stored in system memory (one scan line at a time).

Option "XaaNoScanlineImageWriteRect"
Disables accelerated transfers of full-color rectangular patterns from system memory to video memory (one scan line at a time).

Option "XaaNoScreenToScreenColorExpandFill"
Disables accelerated rectangular expansion blits from source patterns stored in offscreen video memory.

Option "XaaNoScreenToScreenCopy"
Disables accelerated copies of rectangular regions from one part of video memory to another part of video memory.

Option "XaaNoSolidBresenhamLine"
Disables accelerated solid Bresenham line draws.

Option "XaaNoSolidFillRect"
Disables accelerated solid-color fills of rectangles.

Option "XaaNoSolidFillTrap"
Disables accelerated solid-color fills of Bresenham trapezoids.

Option "XaaNoSolidHorVertLine"
Disables accelerated solid horizontal and vertical line draws.

Option "XaaNoSolidTwoPointLine"
Disables accelerated solid line draws between two arbitrary points.

Each **Screen** section may optionally contain one or more **Display** subsections. Those subsections provide depth/fbbpp specific configuration information, and the one chosen depends on the depth and/or fbbpp that is being used for the screen. The **Display** subsection format is described in the section below.

DISPLAY SUBSECTION

Each **Screen** section may have multiple **Display** subsections. The "active" **Display** subsection is the first that matches the depth and/or fbbpp values being used, or failing that, the first that has neither a depth or fbbpp value specified. The **Display** subsections are optional. When there isn't one that matches the depth and/or fbbpp values being used, all the parameters that can be specified here fall back to their defaults.

Display subsections have the following format:

> **SubSection "Display"**
> **Depth** *depth*
> *entries*
> ...
> **EndSubSection**

Depth *depth*
This entry specifies what colour depth the **Display** subsection is to be used for. This entry is usually specified, but it may be omitted to create a match-all **Display** subsection or when wishing to match only against the **FbBpp** parameter. The range of *depth* values that are allowed depends on the driver. Most driver support 8, 15, 16 and 24. Some also support 1 and/or 4, and some may support other values (like 30). Note: *depth* means the number of bits in a pixel that are actually used to determine the pixel colour. 32 is not a valid *depth* value. Most hardware that uses 32 bits per pixel only uses 24 of them to hold the colour information, which means that the colour depth is 24, not 32.

FbBpp *bpp*
This entry specifies the framebuffer format this **Display** subsection is to be used for. This entry is only needed when providing depth 24 configurations that allow a choice between a 24 bpp packed framebuffer format and a 32bpp sparse framebuffer format. In most cases this entry should not be used.

Weight *red-weight green-weight blue-weight*
This optional entry specifies the relative RGB weighting to be used for a screen is being used at depth 16 for drivers that allow multiple formats. This may also be specified from the command line with the **−weight** option (see XFree86(1)).

Virtual *xdim ydim*
This optional entry specifies the virtual screen resolution to be used. *xdim* must be a multiple of either 8 or 16 for most drivers, and a multiple of 32 when running in monochrome mode. The given value will be rounded down if this is not the case. Video modes which are too large for the specified virtual size will be rejected. If this entry is not present, the virtual screen resolution will be set to accommodate all the valid video modes given in the **Modes** entry. Some drivers/hardware combinations do not support virtual screens. Refer to the appropriate driver-specific documentation for details.

ViewPort *x0 y0*
This optional entry sets the upper left corner of the initial display. This is only relevant when the virtual screen resolution is different from the resolution of the initial video mode. If this entry is not given, then the initial display will be centered in the virtual display area.

Modes "*mode-name***"** *...*
This optional entry specifies the list of video modes to use. Each *mode-name* specified must be in double quotes. They must correspond to those specified or referenced in the appropriate **Monitor** section (including implicitly referenced built-in VESA standard modes). The server will delete modes from this list which don't satisfy various requirements. The first valid mode in this list will be the default display mode for startup. The list of valid modes is converted internally into a circular list. It is possible to switch to the next mode with **Ctrl+Alt+Keypad-Plus** and to the previous mode with **Ctrl+Alt+Keypad-Minus**. When this entry is omitted, the valid modes referenced by the appropriate **Monitor** section will be used. If the **Monitor** section contains no modes, then the selection will be taken from the built-in VESA standard modes.

Visual "*visual-name***"**
This optional entry sets the default root visual type. This may also be specified from the command line (see the Xserver(1) man page). The visual types available for depth 8 are (default is

PseudoColor):

StaticGray
GrayScale
StaticColor
PseudoColor
TrueColor
DirectColor

The visual type available for the depths 15, 16 and 24 are (default is **TrueColor**):

TrueColor
DirectColor

Not all drivers support **DirectColor** at these depths.

The visual types available for the depth 4 are (default is **StaticColor**):

StaticGray
GrayScale
StaticColor
PseudoColor

The visual type available for the depth 1 (monochrome) is **StaticGray**.

Black *red green blue*
This optional entry allows the "black" colour to be specified. This is only supported at depth 1. The default is black.

White *red green blue*
This optional entry allows the "white" colour to be specified. This is only supported at depth 1. The default is white.

Options
Option flags may be specified in the **Display** subsections. These may include driver-specific options and driver-independent options. The former are described in the driver-specific documentation. Some of the latter are described above in the section about the **Screen** section, and they may also be included here.

SERVERLAYOUT SECTION

The config file may have multiple **ServerLayout** sections. A "server layout" represents the binding of one or more screens (**Screen** sections) and one or more input devices (**InputDevice** sections) to form a complete configuration. In multi-head configurations, it also specifies the relative layout of the heads. A **ServerLayout** section is considered "active" if it is referenced by the **–layout** command line option or by an **Option "DefaultServerLayout"** entry in the **ServerFlags** section (the former takes precedence over the latter). If those options are not used, the first **ServerLayout** section found in the config file is considered the active one. If no **ServerLayout** sections are present, the single active screen and two active (core) input devices are selected as described in the relevant sections above.

ServerLayout sections have the following format:

Section "ServerLayout"
  Identifier **"***name***"**
  Screen **"***screen-id***"**
  ...
  InputDevice **"***idev-id***"**
  ...
  options
  ...
EndSection

Each **ServerLayout** section must have an **Identifier** entry and at least one **Screen** entry.

The **Identifier** entry specifies the unique name for this server layout. The **ServerLayout** section provides information specific to the whole session, including session-specific **Options**. The **ServerFlags** options (described above) may be specified here, and ones given here override those given in the **ServerFlags** section.

The entries that may be used in this section are described here.

Screen *screen-num* **"screen-id"** *position-information*
One of these entries must be given for each screen being used in a session. The *screen-id* field is mandatory, and specifies the **Screen** section being referenced. The *screen-num* field is optional, and may be used to specify the screen number in multi-head configurations. When this field is omitted, the screens will be numbered in the order that they are listed in. The numbering starts from 0, and must be consecutive. The *position-information* field describes the way multiple screens are positioned. There are a number of different ways that this information can be provided:

x y

Absolute *x y*
These both specify that the upper left corner's coordinates are (*x*,*y*). The **Absolute** keyword is optional. Some older versions of XFree86 (4.2 and earlier) don't recognise the **Absolute** keyword, so it's safest to just specify the coordinates without it.

RightOf **"***screen-id***"**

LeftOf **"***screen-id***"**

Above **"***screen-id***"**

Below **"***screen-id***"**

Relative **"***screen-id***"** *x y*
These give the screen's location relative to another screen. The first four position the screen immediately to the right, left, above or below the other screen. When positioning to the right or left, the top edges are aligned. When positioning above or below, the left edges are aligned. The **Relative** form specifies the offset of the screen's origin (upper left corner) relative to the origin of another screen.

InputDevice **"***idev-id***"** **"***option***"** *...*
One of these entries should be given for each input device being used in a session. Normally at least two are required, one each for the core pointer and keyboard devices. If either of those is missing, suitable **InputDevice** entries are searched for using the method described above in the **INPUTDEVICE** section. The *idev-id* field is mandatory, and specifies the name of the **InputDevice** section being referenced. Multiple *option* fields may be specified, each in double quotes. The options permitted here are any that may also be given in the **InputDevice** sections. Normally only session-specific input device options would be used here. The most commonly used options are:

"CorePointer"
"CoreKeyboard"
"SendCoreEvents"

and the first two should normally be used to indicate the core pointer and core keyboard devices respectively.

Options
Any option permitted in the **ServerFlags** section may also be specified here. When the same option appears in both places, the value given here overrides the one given in the **ServerFlags** section.

Here is an example of a **ServerLayout** section for a dual headed configuration with two mice:

```
Section "ServerLayout"
  Identifier "Layout 1"
```

```
    Screen      "MGA 1"
    Screen      "MGA 2" RightOf "MGA 1"
    InputDevice "Keyboard 1" "CoreKeyboard"
    InputDevice "Mouse 1"    "CorePointer"
    InputDevice "Mouse 2"    "SendCoreEvents"
    Option      "BlankTime"  "5"
EndSection
```

DRI SECTION

This optional section is used to provide some information for the Direct Rendering Infrastructure. Details about the format of this section can be found in the README.DRI document, which is also available online at <*http://www.xfree86.org/current/DRI.html*>.

VENDOR SECTION

The optional **Vendor** section may be used to provide vendor-specific configuration information. Multiple **Vendor** sections may be present, and they may contain an **Identifier** entry and multiple **Option** flags. The data therein is not used in this release.

FILES

For an example of an XF86Config file, see the file installed as /usr/X11R6/lib/X11/XF86Config.eg.

SEE ALSO

X(7), Xserver(1), XFree86(1), apm(4), chips(4), cirrus(4), cyrix(4), fbdev(4), glide(4), glint(4), i128(4), i740(4), i810(4), imstt(4), mga(4), neomagic(4), nv(4), r128(4), rendition(4), savage(4), s3virge(4), siliconmotion(4), sis(4), sunbw2(4), suncg14(4), suncg3(4), suncg6(4), sunffb(4), sunleo(4), suntcx(4), tdfx(4), tga(4), trident(4), tseng(4), v4l(4), vesa(4), vga(4), vmware(4),
README <*http://www.xfree86.org/current/README.html*>,
RELNOTES <*http://www.xfree86.org/current/RELNOTES.html*>,
README.mouse <*http://www.xfree86.org/current/mouse.html*>,
README.DRI <*http://www.xfree86.org/current/DRI.html*>,
Status <*http://www.xfree86.org/current/Status.html*>,
Install <*http://www.xfree86.org/current/Install.html*>.

AUTHORS

This manual page was largely rewritten for XFree86 4.0 by David Dawes <*dawes@xfree86.org*>.

NAME

xf86config − generate an XF86Config file

SYNOPSIS

xf86config

DESCRIPTION

xf86config is an interactive program for generating an XF86Config file for use with XFree86 X servers.

Note that the default name used by *xf86config* for the XF86Config file is system-dependent. For instance, on some systems, XF86Config-4 is used, and on OS/2, XConfig is used.

FILES

/usr/X11R6/lib/X11/Cards
: Video cards database

SEE ALSO

XFree86(1), XF86Config(5), reconfig(1)

AUTHOR

Harm Hanemaayer.

NAME

xf86cfg - Graphical configuration tool for XFree86 4.0

SYNOPSIS

xf86cfg [-xf86config *XF86Config*] [-modulepath *moduledir*] [-fontpath *fontsdir*] [-toolkitoption ...]

DESCRIPTION

Xf86cfg is a tool to configure *XFree86 4.0*, and can be used to either write the initial configuration file or make customizations to the current configuration.

When the **DISPLAY** environment variable is not set, xf86cfg will run the command *XFree86 -configure* to allow the xserver detect the hardware in the computer, and write an initial *XF86Config* file in the user's home directory. Then, it will start XFree86 and allow customizations.
If the **DISPLAY** environment variable is set, xf86cfg will read the default *XF86Config*, that may not be the same being used by the current server, and allow customizations.

To use an alternative location for modules or fonts the respective search paths may be specified.

Unless there is an **Apply** button in the current xf86cfg dialog, the changes made will take place the next time *XFree86* is started.

Xf86cfg allows addition and configuration of new devices, such as video cards, monitors, keyboards and mouses.

Screen layout configuration for xinerama or traditional multi-head is also available.

Modelines can be configured or optimized.

AccessX basic configurations can be made in the xf86cfg's accessx section.

OPTIONS

-xf86config
: Specifies an alternate XF86Config file for configuration.

-modulepath
: Specifies where xf86cfg, and the server it may start, should look for XFree86 modules.

-serverpath
: Specifies the complete path, not including the binary name, of the XFree86 binary.

-fontpath
: Specifies the path to the fonts that should be used by the server started by xf86cfg.

-rgbpath
: Specifies the path to the rgb.txt file that should be used by the server started by xf86cfg, if any.

-textmode
: If xf86cfg was compiled with support to ncurses, this option makes xf86cfg enters a text mode interface.

-nomodules
: When built with support for loading modules, this options changes xf86cfg behaviour, so that it will not load any modules, and thus start quicker.

ENVIRONMENT

DISPLAY
: Default host and display number

XWINHOME
: Directory where XFree86 was installed, defaults to /usr/X11R6.

XENVIRONMENT
: Name of a resource file that overrides the global resources stored in the RESOURCE_MANAGER property

FILES

/etc/XF86Config
Server configuration file

/etc/X11/XF86Config
Server configuration file

/usr/X11R6/etc/XF86Config
Server configuration file

/usr/X11R6/lib/X11/XF86Config.hostname
Server configuration file

/usr/X11R6/lib/X11/XF86Config
Server configuration file

/usr/X11R6/lib/X11/app-default/XF86Cfg
Specifies xf86cfg resources

/usr/X11R6/lib/X11/xkb/X0-config.keyboard
Keyboard specific configuration

SEE ALSO

XFree86(1) *XF86Config*(5)

COPYRIGHT

AUTHORS

Paulo César Pereira de Andrade *<pcpa@conectiva.com.br>*
The XFree86 Project

BUGS

Probably.

NAME

reconfig − convert old Xconfig to new XF86Config

SYNOPSIS

reconfig < *Xconfig* > *XF86Config*

DESCRIPTION

The *reconfig* program converts the Xconfig file format used in XFree86 versions prior to 3.1 into the XF86Config format currently used. The XF86Config format contains more information than the Xconfig format, so manual editing is required after converting.

SEE ALSO

XFree86(1), XF86Config(4/5), xf86config(1)

AUTHOR

Gertjan Akkerman.

BUGS

Comment lines are stripped out when converting.

NAME

getconfig - get configuration information for the XFree86 server

SYNOPSIS

getconfig [*option* ...]

DESCRIPTION

getconfig is a programatic interface that is used by the **XFree86** server to get configuration information about video hardware when operating without an **XF86Config** file.

This implementation of **getconfig** is written in perl. It processes a prioritized and ordered list of rules supplied internally and from meta-configuration files. The rules are in the form of perl expressions. **getconfig** writes to standard output the XF86Config-style configuration data specified by the last highest priority rule that evaluates to true. Information about the format of the meta-configuration files can be found in the getconfig(5) manual page.

OPTIONS

–I *search-path*
: Specify the search path to use for meta-config files. *search-path* is a comma-separated list of directories to search. Each directory in the search path is searched for files with a *.cfg* suffix. Each such file is opened and checked for a valid signature string. Rules are read from files with a valid signature string and appended to the list of rules to evaluate. If no search path is specified, only the internally supplied configuration rules will be used.

–D Enable debugging output.

–V Print out the version information and exit.

–X *XFree86-version*
: Specify the XFree86 version in numeric (integer) form.

–b *subsys-id*
: Specify the PCI subsystem ID of the video device.

–c *class* Specify the PCI class of the video device.

–d *device-id*
: Specify the PCI devide ID of the video device.

–r *revision*
: Specify the PCI revision of the video device.

–s *subsysvendor-id*
: Specify the PCI subsystem vendor ID of the video device.

–v *vendor-id*
: Specify the PCI vendor ID of the video device.

FILES

.cfg files located in the search path. The search path typically specified by the **XFree86** server is:

/etc/X11
/usr/X11R6/etc/X11
<modulepath>
/usr/X11R6/lib/X11/getconfig

where *<modulepath>* is the **XFree86** server's module search path.

SEE ALSO

getconfig(5), XFree86(1), XF86Config(5).

AUTHORS

The XFree86 automatic configuration support and the **getconfig** interface was written by David H. Dawes, with the support of X-Oz Technologies.

NAME

getconfig - meta configuration files for getconfig(1)

SYNOPSIS

∗.cfg

DESCRIPTION

getconfig is a programatic interface that is used by the **XFree86** server to get configuration information about video hardware when operating without an **XF86Config** file.

This implementation of **getconfig** is written in perl. It processes rules from meta-configuration files. All meta-configuration files have a *.cfg* suffix.

Lines starting with a pound-sign (#) are comments, and are ignored. Blank lines that consist only of white space are also treated as comments and ignored.

The first non-comment line must be a signature string followed by the file format version number. The signature string is

"XFree86 Project getconfig rules file. Version: "

The currently defined version is "1.0". Files that do not have the correct signature string are ignored.

The remaining non-comment lines define rules. The start of a new rule is indicated by a line with no leading white space. Subsequent lines making up a rule must be indented with white space. Logical lines within a rule may be split over multiple physical lines by using the usual continuation convention ('\' at the end of the line). The first logical line of each rule is a perl expression. It may be any valid perl expression whose evaluated (with 'eval') result may be used as the argument to a perl 'if' statement. The second logical line should be the name of the XFree86 video driver to use when the rule is true, and subsequent logical lines of each rule, if present, are additional configuration output for the video device's **XF86Config Device** section. The driver name and additional lines of configuration information are written to standard output when the rule is chosen as the successful rule.

Pseudo rules consisting of perl expressions may be present in the file for the purpose of defining custom perl variables or setting the weight to use for the following rules. Pseudo rules are rules that consist of a single logical line only, and the are never candidates themselves for the successful rule.

Several perl variables are pre-defined, and may be used within rules. They include:

$vendor	PCI vendor ID
$device	PCI device ID
$revision	PCI revision ID
$subsys	PCI subsystem ID
$subsysVendor	PCI subsystem vendor ID
$class	PCI class
$XFree86Version	XFree86 version, as a 'v' string
$XFree86VersionNumeric	XFree86 numeric version
$XFree86VersionMajor	XFree86 major version
$XFree86VersionMinor	XFree86 minor version
$XFree86VersionPatch	XFree86 patch version
$XFree86VersionSnap	XFree86 snap version
$weight	current rule weight

The **$weight** variable deterines the weight of the rules as they are processed. The weight for subsequent rules may be set with a pseudo rule that sets or changes the value of **$weight**. The default weight, and the weight used for built-in rules is 500. The meta-configuration files are processed in an unpredictable order. The weighting of the rules is used to determine their relative priority

After processing all of the rules, both built-in and those read from the meta-configration files, the **getconfig** program chooses as the successful rule the last and highest weighted rule that evaluates to true.

FILES

.cfg files located in the search path. The search path typically specified by the **XFree86** server is:

/etc/X11
/usr/X11R6/etc/X11
<modulepath>
/usr/X11R6/lib/X11/getconfig

where *<modulepath>* is the **XFree86** server's module search path.

/usr/X11R6/lib/X11/getconfig/xfree86.cfg
Default rules file that gets installed. This file doesn't contain any rules by default.

/usr/X11R6/lib/X11/getconfig/cfg.sample
A sample rules file that gives some examples of what types of rules can appear in rules files.

SEE ALSO

getconfig(1), XFree86(1), XF86Config(5).

AUTHORS

The XFree86 automatic configuration support and the **getconfig** interface was written by David H. Dawes, with the support of X-Oz Technologies.

NAME

apm − Alliance ProMotion video driver

SYNOPSIS

Section "Device"
Identifier "*devname***"**
Driver "apm"
...
EndSection

DESCRIPTION

apm is an XFree86 driver for Alliance ProMotion video cards. The driver is accelerated for supported hardware/depth combination. It supports framebuffer depths of 8, 15, 16, 24 and 32 bits. For 6420, 6422, AT24, AT3D and AT25, all depths are fully accelerated except 24 bpp for which only screen to screen copy and rectangle filling is accelerated.

SUPPORTED HARDWARE

The **apm** driver supports PCI and ISA video cards on the following Alliance ProMotion chipsets

ProMotion 6420

ProMotion 6422

AT24

AT3D

AT25

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details. This section only covers configuration details specific to this driver.

The driver auto-detects the chipset type, but the following **ChipSet** names may optionally be specified in the config file **"Device"** section, and will override the auto-detection:

"6422", "at24", "at3d".

The AT25 is Chipset "at3d" and the 6420 is 6422.

The driver will auto-detect the amount of video memory present for all chips. The actual amount of video memory can also be specified with a **VideoRam** entry in the config file **"Device"** section.

The following driver **Options** are supported:

Option "HWCursor" "*boolean***"**
Enable or disable the hardware cursor. Default: on.

Option "NoAccel" "*boolean***"**
Disable or enable acceleration. Default: acceleration is enabled.

Option "NoLinear" "*boolean***"**
Disable or enable use of linear frame buffer. Default: on. Note: it may or may not work. Tell me if you need it.

Option "PciRetry" "*boolean***"**
Enable or disable PCI retries. Default: off.

Option "Remap_DPMS_On" "*string***"**

Option "Remap_DPMS_Standby" "*string***"**

Option "Remap_DPMS_Suspend" "*string***"**

Option "Remap_DPMS_Off" "*string***"**
Remaps the corresponding DPMS events. I've found that my Hercules 128/3D swaps Off and Suspend events. You can correct that with
Option "Remap_DPMS_Suspend" "Off"

Option "Remap_DPMS_Off" "Suspend"

in the **Device** section of the config file.

Option "SWCursor" "*boolean***"**
Force the software cursor. Default: off.

Option "ShadowFB" "*boolean***"**
Enable or disable use of the shadow framebuffer layer. Default: off.

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7)

AUTHORS

Authors include: ...

NAME

ati − ATI video driver

SYNOPSIS

Section "Device"
Identifier "*devname***"**
Driver "ati"
...
EndSection

DESCRIPTION

ati is an XFree86 driver for ATI video cards. THIS MAN PAGE NEEDS TO BE FILLED IN.

SUPPORTED HARDWARE

The **ati** driver supports...

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details. This section only covers configuration details specific to this driver.

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7)

AUTHORS

Authors include: ...

NAME

r128 − ATI Rage 128 video driver

SYNOPSIS

Section "Device"
 Identifier "*devname***"**
 Driver "r128"
 ...
EndSection

DESCRIPTION

r128 is an XFree86 driver for ATI Rage 128 based video cards. It contains full support for 8, 15, 16 and 24 bit pixel depths, hardware acceleration of drawing primitives, hardware cursor, video modes up to 1800x1440 @ 70Hz, doublescan modes (e.g., 320x200 and 320x240), gamma correction at all pixel depths, a fully programming dot clock and robust text mode restoration for VT switching.

SUPPORTED HARDWARE

The **r128** driver supports all ATI Rage 128 based video cards including the Rage Fury AGP 32MB, the XPERT 128 AGP 16MB and the XPERT 99 AGP 8MB.

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details. This section only covers configuration details specific to this driver.

The driver auto-detects all device information necessary to initialize the card. However, if you have problems with auto-detection, you can specify:

```
VideoRam - in kilobytes
MemBase  - physical address of the linear framebuffer
IOBase   - physical address of the MMIO registers
ChipID   - PCI DEVICE ID
```

In addition, the following driver **Options** are supported:

Option "SWcursor" "*boolean***"**
Selects software cursor. The default is **off.**

Option "NoAccel" "*boolean***"**
Enables or disables all hardware acceleration. The default is to **enable** hardware acceleration.

Option "Dac6Bit" "*boolean***"**
Enables or disables the use of 6 bits per color component when in 8 bpp mode (emulates VGA mode). By default, all 8 bits per color component are used. The default is **off.**

Option "VideoKey" "*integer***"**
This overrides the default pixel value for the YUV video overlay key. The default value is **undefined.**

Option "Display" "*string***"**
Select display mode for devices which support flat panels. Supported modes are:

"FP" - use flat panel;

"CRT" - use cathode ray tube;

"Mirror" - use both FP and CRT;

"BIOS" - use mode as configured in the BIOS.

The default is **FP.**

The following **Options** are mostly important for non-x86 architectures:

Option "ProgramFPRegs" "*boolean***"**
Enable or disable programming of the flat panel registers. Beware that this may damage your panel, so use this **at your own risk.** The default depends on the device.

Option "PanelWidth" "*integer***"**

Option "PanelHeight" "*integer***"**
Override the flat panel dimensions in pixels. They are used to program the flat panel registers and normally determined using the video card BIOS. If the wrong dimensions are used, the system may hang.

Option "UseFBDev" "*boolean***"**
Enable or disable use of an OS-specific framebuffer device interface (which is not supported on all OSs). See fbdevhw(4) for further information. Default: off.

Option "DMAForXv" "*boolean***"**
Try or don't try to use DMA for Xv image transfers. This will reduce CPU usage when playing big videos like DVDs, but may cause instabilities. Default: off.

The following additional **Options** are supported:

Option "ShowCache" "*boolean***"**
Enable or disable viewing offscreen cache memory. A development debug option. Default: off.

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7)

AUTHORS

Rickard E. (Rik) Faith *faith@precisioninsight.com*
Kevin E. Martin *kevin@precisioninsight.com*

NAME

radeon − ATI RADEON video driver

SYNOPSIS

Section "Device"
Identifier "*devname***"**
Driver "radeon"
...
EndSection

DESCRIPTION

radeon is a XFree86 driver for ATI RADEON based video cards. It contains full support for 8, 15, 16 and 24 bit pixel depths, dual-head setup, flat panel, hardware 2D acceleration, hardware 3D acceleration (except R300 and IGP series cards), hardware cursor, XV extension, Xinerama extension.

SUPPORTED HARDWARE

The **radeon** driver supports PCI and AGP video cards based on the following ATI chips

R100	Radeon 7200
RV100	Radeon 7000(VE), M6
RS100	Radeon IGP320(M) (2D only)
RV200	Radeon 7500, M7, FireGL 7800
RS200	Radeon IGP330(M)/IGP340(M) (2D only)
RS250	Radeon Mobility 7000 IGP (2D only)
R200	Radeon 8500, 9100, FireGL 8800/8700
RV250	Radeon 9000PRO/9000, M9
RS300	Radeon 9100 IGP (2D only)
RV280	Radeon 9200PRO/9200/9200SE, M9+
R300	Radeon 9700PRO/9700/9500PRO/9500/9600TX, FireGL X1/Z1 (2D only)
R350	Radeon 9800PRO/9800SE/9800, FireGL X2 (2D only)
R360	Radeon 9800XT (2d only)
RV350	Radeon 9600PRO/9600SE/9600, M10/M11, FireGL T2 (2D only)
RV360	Radeon 9600XT (2d only)

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details. This section only covers configuration details specific to this driver.

The driver auto−detects all device information necessary to initialize the card. However, if you have problems with auto−detection, you can specify:

VideoRam − in kilobytes
MemBase − physical address of the linear framebuffer
IOBase − physical address of the MMIO registers
ChipID − PCI DEVICE ID

In addition, the following driver **Options** are supported:

Option "SWcursor" "*boolean***"**
Selects software cursor. The default is **off.**

Option "NoAccel" "*boolean***"**
Enables or disables all hardware acceleration.
The default is to **enable** hardware acceleration.

Option "Dac6Bit" "*boolean*"
Enables or disables the use of 6 bits per color component when in 8 bpp mode (emulates VGA mode). By default, all 8 bits per color component are used.
The default is **off.**

Option "VideoKey" "*integer*"
This overrides the default pixel value for the YUV video overlay key.
The default value is **0x1E.**

Option "UseFBDev" "*boolean*"
Enable or disable use of an OS−specific framebuffer device interface (which is not supported on all OSs). See fbdevhw(4) for further information.
The default is **off.**

Option "AGPMode" "*integer*"
Set AGP data transfer rate. (used only when DRI is enabled)
1 −− x1 (default)
2 −− x2
4 −− x4
others −− invalid

Option "AGPFastWrite" "*boolean*"
Enable AGP fast write.
(used only when DRI is enabled)
The default is **off.**

Option "BusType" "*string*"
Used to replace previous ForcePCIMode option. Should only be used when driver's bus detection is incorrect or you want to force a AGP card to PCI mode. Should NEVER force a PCI card to AGP bus.
PCI −− PCI bus
AGP −− AGP bus
PCIE −− PCI Express (falls back to PCI at present)
(used only when DRI is enabled)
The default is **auto detect.**

Option "ForcePCIMode" "*boolean*"
Force to use PCI GART for DRI acceleration. This option is deprecated in favor of the **BusType** option above and will be removed in the next release.

Option "DDCMode" "*boolean*"
Force to use the modes queried from the connected monitor.
The default is **off.**

Option "DisplayPriority" "*string*"
Used to prevent flickering or tearing problem caused by display buffer underflow.
AUTO −− Driver calculated (default).
BIOS −− Remain unchanged from BIOS setting.
Use this if the calculation is not correct
for your card.
HIGH −− Force to the highest priority.
Use this if you have problem with above options.
This may affect performence slightly.
The default value is **AUTO.**

Option "MonitorLayout" "*string*"
This option is used to overwrite the detected monitor types. This is only required when driver makes a false detection. The possible monitor types are:
NONE −− Not connected
CRT −− Analog CRT monitor

TMDS −− Desktop flat panel
LVDS −− Laptop flat panel
This option can be used in following format:
Option "MonitorLayout" "[type on primary], [type on secondary]"
For example, Option "MonitorLayout" "CRT, TMDS"

Primary/Secondary head for dual−head cards:
(when only one port is used, it will be treated as the primary regardless)
Primary head:
DVI port on DVI+VGA cards
LCD output on laptops
Internal TMDS prot on DVI+DVI cards
Secondary head:
VGA port on DVI+VGA cards
VGA port on laptops
External TMDS port on DVI+DVI cards

The default value is **undefined.**

Option "CloneMode" "*string*"
Set the first mode for the secondary head. It can be different from the modes used for the primary head. If you don't have this line while clone is on, the modes specified for the primary head will be used for the secondary head.
For example, Option "CloneMode" "1024x768"
The default value is **undefined.**

Option "CloneHSync" "*string*"
Set the horizontal sync range for the secondary monitor. It is not required if a DDC−capable monitor is connected.
For example, Option "CloneHSync" "30.0-86.0"
The default value is **undefined.**

Option "CloneVRefresh" "*string*"
Set the vertical refresh range for the secondary monitor. It is not required if a DDC−capable monitor is connected.
For example, Option "CloneVRefresh" "50.0-120.0"
The default value is **undefined.**

Option "OverlayOnCRTC2" "*boolean*"
Force hardware overlay to clone head.
The default value is **off.**

Option "IgnoreEDID" "*boolean*"
Do not use EDID data for mode validation, but DDC is still used for monitor detection. This is different from NoDDC option.
The default value is **off.**

Option "PanelSize" "*string*"
Should only be used when driver cannot detect the correct panel size. Apply to both desktop (TMDS) and laptop (LVDS) digital panels. When a valid panel size is specified, the timings collected from DDC and BIOS will not be used. If you have a panel with timings different from that of a standard VESA mode, you have to provide this information through the Modeline.
For example, Option "PanelSize" "1400x1050"
The default value is **none.**

Option "PanelOff" "*boolean*"
Disable panel output. Only used when clone is enabled.
The default value is **off.**

Option "EnablePageFlip" "*boolean*"
Enable page flipping for 3D acceleration. This will increase performance but not work correctly in some rare cases, hence the default is **off.**

Option "ForceMinDotClock" "*frequency*"
Override minimum dot clock. Some Radeon BIOSes report a minimum dot clock unsuitable (too high) for use with television sets even when they actually can produce lower dot clocks. If this is the case you can override the value here. **Note that using this option may damage your hardware.** You have been warned. The **frequency** parameter may be specified as a float value with standard suffixes like "k", "kHz", "M", "MHz".

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7)

AUTHORS

Authors include: ...

NAME

chips − Chips and Technologies video driver

SYNOPSIS

Section "Device"
Identifier "*devname***"**
Driver "chips"
...
EndSection

DESCRIPTION

chips is an XFree86 driver for Chips and Technologies video processors. The majority of the Chips and Technologies chipsets are supported by this driver. In general the limitation on the capabilities of this driver are determined by the chipset on which it is run. Where possible, this driver provides full acceleration and supports the following depths: 1, 4, 8, 15, 16, 24 and on the latest chipsets an 8+16 overlay mode. All visual types are supported for depth 1, 4 and 8 and both TrueColor and DirectColor visuals are supported where possible. Multi-head configurations are supported on PCI or AGP buses.

SUPPORTED HARDWARE

The **chips** driver supports video processors on most of the bus types currently available. The chipsets supported fall into one of three architectural classes. A **basic** architecture, the **WinGine** architecture and the newer **HiQV** architecture.

Basic Architecture

The supported chipsets are **ct65520, ct65525, ct65530, ct65535, ct65540, ct65545, ct65546** and **ct65548**

Color depths 1, 4 and 8 are supported on all chipsets, while depths 15, 16 and 24 are supported only on the **65540, 65545, 65546** and **65548** chipsets. The driver is accelerated when used with the **65545, 65546** or **65548** chipsets, however the DirectColor visual is not available.

Wingine Architecture

The supported chipsets are **ct64200** and **ct64300**

Color depths 1, 4 and 8 are supported on both chipsets, while depths 15, 16 and 24 are supported only on the **64300** chipsets. The driver is accelerated when used with the **64300** chipsets, however the DirectColor visual is not available.

HiQV Architecture

The supported chipsets are **ct65550, ct65554, ct65555, ct68554, ct69000** and **ct69030**

Color depths 1, 4, 8, 15, 16, 24 and 8+16 are supported on all chipsets. The DirectColor visual is supported on all color depths except the 8+16 overlay mode. Full acceleration is supplied for all chipsets.

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details. This section only covers configuration details specific to this driver.

The driver auto-detects the chipset type, but the following **ChipSet** names may optionally be specified in the config file **"Device"** section, and will override the auto-detection:

> "ct65520", "ct65525", "ct65530", "ct65535", "ct65540", "ct65545", "ct65546", "ct65548", "ct65550", "ct65554", "ct65555", "ct68554", "ct69000", "ct69030", "ct64200", "ct64300".

The driver will auto-detect the amount of video memory present for all chipsets. But maybe overridden with the **VideoRam** entry in the config file **"Device"** section.

The following driver **Options** are supported, on one or more of the supported chipsets:

Option "NoAccel" "*boolean***"**
Disable or enable acceleration. Default: acceleration is enabled.

Option "NoLinear" "*boolean*"
Disables linear addressing in cases where it is enabled by default. Default: off

Option "Linear" "*boolean*"
Enables linear addressing in cases where it is disabled by default. Default: off

Option "HWCursor" "*boolean*"
Enable or disable the HW cursor. Default: on.

Option "SWCursor" "*boolean*"
Enable or disable the HW cursor. Default: off.

Option "STN" "*boolean*"
Force detection of STN screen type. Default: off.

Option "UseModeline" "*boolean*"
Reprogram flat panel timings with values from the modeline. Default: off

Option "FixPanelSize" "*boolean*"
Reprogram flat panel size with values from the modeline. Default: off

Option "NoStretch" "*boolean*"
This option disables the stretching on a mode on a flat panel to fill the screen. Default: off

Option "LcdCenter" "*boolean*"
Center the mode displayed on the flat panel on the screen. Default: off

Option "HWclocks" "*boolean*"
Force the use of fixed hardware clocks on chips that support both fixed and programmable clocks. Default: off

Option "UseVclk1" "*boolean*"
Use the Vclk1 programmable clock on **HiQV** chipsets instead of Vclk2. Default: off

Option "FPClock8" "*float*"

Option "FPClock16" "*float*"

Option "FPClock24" "*float*"

Option "FPClock32" "*float*"
Force the use of a particular video clock speed for use with the flat panel at a specified depth

Option "MMIO" "*boolean*"
Force the use of memory mapped IO for acceleration registers. Default: off

Option "FullMMIO" "*boolean*"
Force the use of memory mapped IO where it can be used. Default: off

Option "SuspendHack" "*boolean*"
Force driver to leave centering and stretching registers alone. This can fix some laptop suspend/resume problems. Default: off

Option "Overlay"
Enable 8+24 overlay mode. Only appropriate for depth 24. Default: off.

Option "ColorKey" "*integer*"
Set the colormap index used for the transparency key for the depth 8 plane when operating in 8+16 overlay mode. The value must be in the range 2−255. Default: 255.

Option "VideoKey" "*integer*"
This sets the default pixel value for the YUV video overlay key. Default: undefined.

Option "ShadowFB" "*boolean*"
Enable or disable use of the shadow framebuffer layer. Default: off.

Option "SyncOnGreen" "*boolean*"
Enable or disable combining the sync signals with the green signal. Default: off.

Option "ShowCache" "*boolean*"
Enable or disable viewing offscreen memory. Used for debugging only Default: off.

Option "18bitBus" "*boolean*"
Force the driver to assume that the flat panel has an 18bit data bus. Default: off.

Option "Crt2Memory" "*integer*"
In a dual-head mode (69030 only) this option selects the amount of memory to set aside for the second head. If not specified, half the memory is used. Default: off.

Option "DualRefresh" "*integer*"
The 69030 supports independent refresh rates on its two display channels. This mode of operations uses additional memory bandwidth and thus limits the maximum colour depth and refresh rate that can be achieved, and so is off by default. Using this option forces the use of an independent refresh rate on the two screens. Default: off.

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7)

You are also recommended to read the README.chips file that comes with all XFree86 distributions, which discusses the **chips** driver in more detail.

AUTHORS

Authors include: Jon Block, Mike Hollick, Regis Cridlig, Nozomi Ytow, Egbert Eich, David Bateman and Xavier Ducoin

NAME

cirrus − Cirrus Logic video driver

SYNOPSIS

Section "Device"
  Identifier "*devname***"**
  Driver "cirrus"
  ...
EndSection

DESCRIPTION

cirrus is an XFree86 driver for Cirrus Logic video chips. THIS MAN PAGE NEEDS TO BE FILLED IN.

SUPPORTED HARDWARE

The **cirrus** driver supports...

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details. This section only covers configuration details specific to this driver.

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7)

AUTHORS

Authors include: ...

NAME

cyrix − Cyrix video driver

SYNOPSIS

Section "Device"
Identifier "*devname***"**
Driver "cyrix"
...
EndSection

DESCRIPTION

cyrix is an XFree86 driver for the Cyrix MediaGX (now Natsemi Geode) series of processors when using the built in video.

SUPPORTED HARDWARE

The **cyrix** driver supports the MediaGX, MediaGXi and MediaGXm processors, as well as the Natsemi 'Geode' branded processors. It supports the CS5510, CS5520, CS5530 and CS5530A companion chips. The driver supports 4, 8, 15 and 16 bit deep displays with video compression and acceleration.

The MediaGX run length compresses its shared framebuffer, for the best performance on a MediaGX machine pick backgrounds that compress well horizonally.

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details. This section only covers configuration details specific to this driver.

The following driver **options** are supported

Option "NoAccel" "*boolean***"**
Disable or enable acceleration. Default: acceleration is enabled.

Option "SWCursor" "*boolean***"**
Disable or enable software cursor. Default software cursor is enabled and a hardware cursor is used.

Option "HWCursor" "*boolean***"**
Disable or enable hardware cursor. Default hardware cursor is disabled.

Option "ShadowFB" "*boolean***"**
Disable or enable shadow frame buffer. The shadow buffer is normally only used when rotating the screen. The default is false.

Option "Rotate" "CW"

Option "Rotate" "CCW"

Rotate the display clockwise or counterclockwise for use on Cyrix based tablet PC systems. This mode is currently unaccelerated.

BUGS

This driver has not been tested on the original 5510 hardware for some considerable time.

8bit mode does not currently work on the CS5510 with external RAMDAC.

The 5530A video overlay facility is not currently supported.

XFree86 uses the MediaGX 'SoftVGA' interface. On a small number of boards this is buggy and may result in strange illegal instruction traps.

Hardware cursors are not currently supported.

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7)

AUTHORS

Authors include: Richard Hecker, Annius Groenink, Dirk Hohndel, The GGI Project, Alan Cox.

NAME

fbdev − video driver for framebuffer device

SYNOPSIS

Section "Device"
 Identifier "*devname***"**
 Driver "fbdev"
 BusID "pci:*bus***:***dev***:***func***"**
 ...
EndSection

DESCRIPTION

fbdev is an XFree86 driver for framebuffer devices. This is a non-accelerated driver, the following framebuffer depths are supported: 8, 15, 16, 24. All visual types are supported for depth 8, and TrueColor visual is supported for the other depths. Multi-head configurations are supported.

SUPPORTED HARDWARE

The **fbdev** driver supports all hardware where a framebuffer driver is available. fbdev uses the os-specific submodule fbdevhw(4) to talk to the kernel device driver. Currently a fbdevhw module is available for linux.

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details. This section only covers configuration details specific to this driver.

For this driver it is not required to specify modes in the screen section of the config file. The **fbdev** driver can pick up the currently used video mode from the framebuffer driver and will use it if there are no video modes configured.

For PCI boards you might have to add a BusID line to the Device section. See above for a sample line. You can use "XFree86 -scanpci" to figure out the correct values.

The following driver **Options** are supported:

Option "fbdev" "*string***"**
The framebuffer device to use. Default: /dev/fb0.

Option "ShadowFB" "*boolean***"**
Enable or disable use of the shadow framebuffer layer. Default: on.

Option "Rotate" "*string***"**
Enable rotation of the display. The supported values are "CW" (clockwise, 90 degrees), "UD" (upside down, 180 degrees) and "CCW" (counter clockwise, 270 degrees). Implies use of the shadow framebuffer layer. Default: off.

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7), fbdevhw(4)

AUTHORS

Authors include: Gerd Knorr, Michel Dänzer, Geert Uytterhoeven

NAME

glide − Glide video driver

SYNOPSIS

Section "Device"
Identifier "*devname***"**
Driver "glide"
...
EndSection

READ THIS IF NOTHING ELSE

This driver has a special requirement that needs to be fulfilled before it will work: You need Glide installed and you need to make a link for the libglide2x.so file. Read the second paragraph in the description below to find out how.

DESCRIPTION

glide is an XFree86 driver for Glide capable video boards (such as 3Dfx Voodoo boards). This driver is mainly for Voodoo 1 and Voodoo 2 boards, later boards from 3Dfx have 2D built-in and you should preferably use a driver separate for those boards or the fbdev(4) driver. This driver is a bit special because Voodoo 1 and 2 boards are very much NOT made for running 2D graphics. Therefore, this driver uses no hardware acceleration (since there is no acceleration for 2D, only 3D). Instead it is implemented with the help of a "shadow" framebuffer that resides entirely in RAM. Selected portions of this shadow framebuffer are then copied out to the Voodoo board at the right time. Because of this, the speed of the driver is very dependent on the CPU. But since the CPU is nowadays actually rather fast at moving data, we get very good speed anyway, especially since the whole shadow framebuffer is in cached RAM.

This driver requires that you have installed Glide. (Which can, at the time of this writing, be found at http://glide.xxedgexx.com/3DfxRPMS.html). Also, you need to tell XFree86 where the libglide2x.so file is placed by making a soft link in the /usr/X11R6/lib/modules directory that points to the libglide2x.so file. For example (if your libglide2x.so file is in /usr/lib):

```
# ln -s /usr/lib/libglide2x.so /usr/X11R6/lib/modules
```

If you have installed /dev/3dfx, the driver will be able to turn on the MTRR registers (through the glide library) if you have a CPU with such registers (see http://glide.xxedgexx.com/MTRR.html). This will speed up copying data to the Voodoo board by as much as 2.7 times and is very noticeable since this driver copies a lot of data... Highly recommended.

This driver supports 16 and 24 bit color modes. The 24 bit color mode uses a 32 bit framebuffer (it has no support for 24 bit packed-pixel framebuffers). Notice that the Voodoo boards can only display 16 bit color, but the shadow framebuffer can be run in 24 bit color. The point of supporting 24 bit mode is that this enables you to run in a multihead configuration with Xinerama together with another board that runs in real 24 bit color mode. (All boards must run the same color depth when you use Xinerama).

Resolutions supported are: 640x480, 800x600, 960x720, 1024x768, 1280x1024 and 1600x1200. Note that not all modes will work on all Voodoo boards. It seems that Voodoo 2 boards support no higher than 1024x768 and Voodoo 1 boards can go to 800x600. If you see a message like this in the output from the server:

```
(EE) GLIDE(0): grSstWinOpen returned ...
```

Then you are probably trying to use a resolution that is supported by the driver but not supported by the hardware.

Refresh rates supported are: 60Hz, 75Hz and 85Hz. The refresh rate used is derived from the normal mode line according to the following table:

Mode-line refresh rate	Used refresh rate
0-74 Hz	60 Hz
74-84 Hz	75 Hz

84- Hz 85 Hz

Thus, if you use a modeline that for example has a 70Hz refresh rate you will only get a 60Hz refresh rate in actuality.

Selecting which Voodoo board to use with the driver is done by using an option called "GlideDevice" in the "Device" section. (If you don't have this option present then the first board found will be selected for that Device section). For example: To use the first Voodoo board, use a "Device" section like this, for example:

```
Section "Device"
  Identifier  "Voodoo"
  Driver      "glide"
  Option      "dpms" "on"
  Option      "GlideDevice" "0"
EndSection
```

And if you have more than one Voodoo board, add another "Device" section with a GlideDevice option with value 1, and so on. (You can use more than one Voodoo board, but SLI configured boards will be treated as a single board.)

Multihead and Xinerama configurations are supported.

Limited support for DPMS screen saving is available. The "standby" and "suspend" modes are just painting the screen black. The "off" mode turns the Voodoo board off and thus works correctly.

This driver does not support a virtual screen size different from the display size.

SUPPORTED HARDWARE

The **glide** driver supports any board that can be used with Glide (such as 3Dfx Voodoo boards)

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details. This section only covers configuration details specific to this driver.

The following driver **Options** are supported:

Option "OnAtExit" "*boolean***"**
: If true, will leave the Voodoo board on when the server exits. Useful in a multihead setup when only the Voodoo board is connected to a second monitor and you don't want that monitor to lose signal when you quit the server. Put this option in the Device section. Default: off.

Option "GlideDevice" "*integer***"**
: Selects which Voodoo board to use. (Or boards, in an SLI configuration). The value should be 0 for the first board, 1 for the second and so on. If it is not present, the first Voodoo board found will be selected. Put this option in the Device section.

EXAMPLE

Here is an example of a part of an XF86Config file that uses a multihead configuration with two monitors. The first monitor is driven by the fbdev video driver and the second monitor is driven by the glide driver.

```
Section "Monitor"
  Identifier    "Monitor 1"
  VendorName    "Unknown"
  ModelName     "Unknown"
  HorizSync     30-70
  VertRefresh   50-80

  # 1024x768 @ 76 Hz, 62.5 kHz hsync
  Modeline "1024x768" 85 1024 1032 1152 1360 768 784 787 823
EndSection

Section "Monitor"
  Identifier    "Monitor 2"
```

```
  VendorName     "Unknown"
  ModelName      "Unknown"
  HorizSync      30-70
  VertRefresh    50-80

  # 1024x768 @ 76 Hz, 62.5 kHz hsync
  Modeline "1024x768" 85 1024 1032 1152 1360 768 784 787 823
EndSection

Section "Device"
  Identifier  "fb"
  Driver      "fbdev"
  Option      "shadowfb"
  Option      "dpms" "on"
  # My video card is on the AGP bus which is usually
  # located as PCI bus 1, device 0, function 0.
  BusID       "PCI:1:0:0"
EndSection

Section "Device"
  # I have a Voodoo 2 board
  Identifier  "Voodoo"
  Driver      "glide"
  Option      "dpms" "on"
  # The next line says I want to use the first board.
  Option      "GlideDevice" "0"
EndSection

Section "Screen"
 Identifier         "Screen 1"
 Device "fb"
 Monitor            "Monitor 1"
 DefaultDepth    16
 Subsection "Display"
  Depth 16
  Modes             "1024x768"
 EndSubSection
EndSection

Section "Screen"
 Identifier         "Screen 2"
 Device "Voodoo"
 Monitor            "Monitor 2"
 DefaultDepth    16
 Subsection "Display"
  Depth 16
  Modes             "1024x768"
 EndSubSection
EndSection

Section "ServerLayout"
 Identifier         "Main Layout"
 # Screen 1 is to the right and screen 2 is to the left
 Screen  "Screen 2"
```

```
  Screen  "Screen 1" "" "" "Screen 2" ""
EndSection
```

If you use this configuration file and start the server with the +xinerama command line option, the two monitors will be showing a single large area where windows can be moved between monitors and overlap from one monitor to the other. Starting the X server with the Xinerama extension can be done for example like this:

```
$ xinit -- +xinerama
```

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7)

AUTHORS

Author: Henrik Harmsen.

NAME

glint – GLINT/Permedia video driver

SYNOPSIS

Section "Device"
Identifier "*devname***"**
Driver "glint"
...
EndSection

DESCRIPTION

glint is an XFree86 driver for 3Dlabs & Texas Instruments GLINT/Permedia based video cards. The driver is rather fully accelerated, and provides support for the following framebuffer depths: 8, 15 (may give bad results with FBDev support), 16, 24 (32 bpp recommended, 24 bpp has problems), 30, and an 8+24 overlay mode.

SUPPORTED HARDWARE

The **glint** driver supports 3Dlabs (GLINT MX, GLINT 500TX, GLINT 300SX, GLINT GAMMA, GLINT DELTA, GLINT GAMMA2, Permedia, Permedia 2, Permedia 2v, Permedia 3, R3, R4) and Texas Instruments (Permedia, Permedia 2) chips.

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details. This section only covers configuration details specific to this driver.

The driver auto-detects the chipset type, but the following **ChipSet** names may optionally be specified in the config file **"Device"** section, and will override the auto-detection:

> "ti_pm2", "ti_pm", "r4", "pm3", "pm2v", "pm2", "pm", "300sx", "500tx", "mx", "gamma", "gamma2", "delta"

The driver will try to auto-detect the amount of video memory present for all chips. If it's not detected correctly, the actual amount of video memory should be specified with a **VideoRam** entry in the config file **"Device"** section.

Additionally, you may need to specify the bus ID of your card with a **BusID** entry in the config file **"Device"** section, especially with FBDev support.

The following driver **Options** are supported:

Option "UseFlatPanel" "*boolean***"**
Enable the FlatPanel feature on the Permedia3. Default: off.

Option "SWCursor" "*boolean***"**
Enable or disable the SW cursor. Default: off. This option disables the **HWCursor** option and vice versa.

Option "NoAccel" "*boolean***"**
Disable or enable acceleration. Default: acceleration is enabled.

Option "Overlay"
Enable 8+24 overlay mode. Only appropriate for depth 24, 32 bpp. (**Note:** This hasn't been tested with FBDev support and probably won't work.) Recognized values are: "8,24", "24,8". Default: off.

Option "PciRetry" "*boolean***"**
Enable or disable PCI retries. (**Note:** This doesn't work with Permedia2 based cards for Amigas.) Default: off.

Option "ShadowFB" "*boolean***"**
Enable or disable use of the shadow framebuffer layer. (**Note:** This disables hardware acceleration.) Default: off.

Option "UseFBDev" "*boolean*"
Enable or disable use of an OS-specific fb interface (which is not supported on all OSs). See fbdevhw(4) for further information. Default: off.

Option "BlockWrite" "*boolean*"
Enable or disable block writes for the various Permedia 2 chips. This improves acceleration in general, but disables it for some special cases. Default: off.

Option "FireGL3000" "*boolean*"
If you have a card of the same name, turn this on. Default: off.

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7)

AUTHORS

Authors include: Alan Hourihane, Dirk Hohndel, Stefan Dirsch, Michel Dänzer, Sven Luther

NAME

i128 − Number 9 I128 video driver

SYNOPSIS

Section "Device"
 Identifier "*devname***"**
 Driver "i128"
 ...
EndSection

DESCRIPTION

i128 is an XFree86 driver for Number 9 I128 video cards. The driver is accelerated and provides support for all versions of the I128 chip family, including the SGI flatpanel configuration. Multi-head configurations are supported.

SUPPORTED HARDWARE

The **i128** driver supports PCI and AGP video cards based on the following I128 chips:

I128 rev 1 (original)

I128-II

I128-T2R Ticket 2 Ride

I128-T2R4 Ticket 2 Ride IV

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details. This section only covers configuration details specific to this driver.

The driver auto-detects the chipset type and may not be overridden.

The driver auto-detects the amount of video memory present for all chips and may not be overridden.

The following driver **Options** are supported:

Option "HWCursor" "*boolean***"**
Enable or disable the HW cursor. Default: on.

Option "NoAccel" "*boolean***"**
Disable or enable acceleration. Default: acceleration is enabled.

Option "SyncOnGreen" "*boolean***"**
Enable or disable combining the sync signals with the green signal. Default: off.

Option "Dac6Bit" "*boolean***"**
Reduce DAC operations to 6 bits. Default: false.

Option "Debug" "*boolean***"**
This turns on verbose debug information from the driver. Default: off.

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7)

AUTHORS

Authors include: Robin Cutshaw (driver), Galen Brooks (flatpanel support).

NAME

i740 − Intel i740 video driver

SYNOPSIS

Section "Device"
Identifier "*devname***"**
Driver "i740"
...
EndSection

DESCRIPTION

i740 is an XFree86 driver for Intel i740 video cards. THIS MAN PAGE NEEDS TO BE FILLED IN.

SUPPORTED HARDWARE

The **i740** driver supports...

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details. This section only covers configuration details specific to this driver.

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7)

AUTHORS

Authors include: ...

NAME

i810 − Intel 8xx integrated graphics chipsets

SYNOPSIS

Section "Device"
Identifier "*devname***"**
Driver "i810"
...
EndSection

DESCRIPTION

i810 is an XFree86 driver for Intel integrated graphics chipsets. The driver supports depths 8, 15, 16 and 24. All visual types are supported in depth 8. For the i810/i815 other depths support the TrueColor and DirectColor visuals. For the 830M and later, only the TrueColor visual is supported for depths greater than 8. The driver supports hardware accelerated 3D via the Direct Rendering Infrastructure (DRI), but only in depth 16 for the i810/i815 and depths 16 and 24 for the 830M and later.

SUPPORTED HARDWARE

i810 supports the i810, i810-DC100, i810e, i815, 830M, 845G, 852GM, 855GM, and 865G chipsets.

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details. This section only covers configuration details specific to this driver.

The Intel 8xx family of integrated graphics chipsets has a unified memory architecture and uses system memory for video ram. For the i810 and i815 familiy of chipset, operating system support for allocating system memory for video use is required in order to use this driver. For the 830M and later, this is required in order for the driver to use more video ram than has been pre-allocated at boot time by the BIOS. This is usually achieved with an "agpgart" or "agp" kernel driver. Linux, and recent versions of FreeBSD, OpenBSD and NetBSD have such kernel drivers available.

By default 8 Megabytes of system memory are used for graphics. For the 830M and later, the default is 8 Megabytes when DRI is not enabled and 32 Megabytes with DRI is enabled. This amount may be changed with the **VideoRam** entry in the config file **Device** section. It may be set to any reasonable value up to 64MB for older chipsets or 128MB for newer chipets. It is advisable to check the XFree86 log file to check if any features have been disabled because of insufficient video memory. In particular, DRI support or tiling mode may be disabled with insufficient video memory. Either of these being disabled will reduce performance for 3D applications. Note however, that increasing this value too much will reduce the amount of system memory available for other applications.

The driver makes use of the video BIOS to program video modes for the 830M and later. This limits the video modes that can be used to those provided by the video BIOS, and to those that will fit into the amount of video memory that the video BIOS is aware of.

The following driver **Options** are supported

Option "NoAccel" "*boolean***"**
Disable or enable acceleration. Default: acceleration is enabled.

Option "SWCursor" "*boolean***"**
Disable or enable software cursor. Default: software cursor is disable and a hardware cursor is used for configurations where the hardware cursor is available.

Option "ColorKey" "*integer***"**
This sets the default pixel value for the YUV video overlay key. Default: undefined.

Option "CacheLines" "*integer***"**
This allows the user to change the amount of graphics memory used for 2D acceleration and video. Decreasing this amount leaves more for 3D textures. Increasing it can improve 2D performance at the expense of 3D performance. Default: depends on the resolution, depth, and available video memory. The driver attempts to allocate at least enough to hold two DVD-sized YUV

buffers by default. The default used for a specific configuration can be found by examining the XFree86 log file.

Option "DRI" "*boolean***"**
Disable or enable DRI support. Default: DRI is enabled for configurations where it is supported.

The following driver **Options** are supported for the i810 and i815 chipsets:

Option "DDC" "*boolean***"**
Disable or enable DDC support. Default: enabled.

Option "Dac6Bit" "*boolean***"**
Enable or disable 6-bits per RGB for 8-bit modes. Default: 8-bits per RGB for 8-bit modes.

Option "XvMCSurfaces" "*integer***"**
This option enables XvMC. The integer parameter specifies the number of surfaces to use. Valid values are 6 and 7. Default: XvMC is disabled.

The following driver **Options** are supported for the 830M and later chipsets:

Option "VBERestore" "*boolean***"**
Enable or disable the use of VBE save/restore for saving and restoring the initial text mode. This is disabled by default because it causes lockups on some platforms. However, there are some cases where it must enabled for the correct restoration of the initial video mode. If you are having a problem with that, try enabling this option. Default: Disabled.

Option "VideoKey" "*integer***"**
This is the same as the **"ColorKey"** option described above. It is provided for compatibility with most other drivers.

Option "XVideo" "*boolean***"**
Disable or enable XVideo support. Default: XVideo is enabled for configurations where it is supported.

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7)

AUTHORS

Authors include: Keith Whitwell, and also Jonathan Bian, Matthew J Sottek, Jeff Hartmann, Mark Vojkovich, Alan Hourihane, H. J. Lu. 830M and 845G support reworked for XFree86 4.3 by David Dawes and Keith Whitwell. 852GM, 855GM, and 865G support added by David Dawes and Keith Whitwell.

NAME

imstt – Integrated Micro Solutions Twin Turbo 128 driver

SYNOPSIS

Section "Device"
Identifier "*devname***"**
Driver "imstt"
...
EndSection

DESCRIPTION

imstt is an XFree86 driver for Integrated Micro Solutions Twin Turbo 128 video chips. THIS MAN PAGE NEEDS TO BE FILLED IN.

SUPPORTED HARDWARE

The **imstt** driver supports...

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details. This section only covers configuration details specific to this driver.

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7)

AUTHORS

Authors include: ...

NAME

mga – Matrox video driver

SYNOPSIS

Section "Device"
Identifier "*devname***"**
Driver "mga"
...
EndSection

DESCRIPTION

mga is an XFree86 driver for Matrox video cards. The driver is fully accelerated, and provides support for the following framebuffer depths: 8, 15, 16, 24, and an 8+24 overlay mode. All visual types are supported for depth 8, and both TrueColor and DirectColor visuals are supported for the other depths except 8+24 mode which supports PseudoColor, GrayScale and TrueColor. Multi-card configurations are supported. XVideo is supported on G200 and newer systems, with either **TexturedVideo** or video overlay. The second head of dual-head cards is supported for the G450 and G550. Support for the second head on G400 cards requires a binary-only "mga_hal" module that is available from Matrox <http://www.matrox.com>, and may be on the CD supplied with the card. That module also provides various other enhancements, and may be necessary to use the DVI (digital) output on the G550 (and other cards).

SUPPORTED HARDWARE

The **mga** driver supports PCI and AGP video cards based on the following Matrox chips:

MGA2064W Millennium (original)

MGA1064SG
Mystique

MGA2164W Millennium II

G100

G200 Millennium G200 and Mystique G200

G400

G450

G550

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details. This section only covers configuration details specific to this driver.

The driver auto-detects the chipset type, but the following **ChipSet** names may optionally be specified in the config file **"Device"** section, and will override the auto-detection:

"mga2064w", "mga1064sg", "mga2164w", "mga2164w agp", "mgag100", "mgag200", "mgag200 pci", "mgag400", "mgag550".

The G450 is Chipset "mgag400" with ChipRev 0x80.

The driver will auto-detect the amount of video memory present for all chips except the Millennium II. In the Millennium II case it defaults to 4096 kBytes. When using a Millennium II, the actual amount of video memory should be specified with a **VideoRam** entry in the config file **"Device"** section.

The following driver **Options** are supported:

Option "ColorKey" "*integer***"**
Set the colormap index used for the transparency key for the depth 8 plane when operating in 8+24 overlay mode. The value must be in the range 2−255. Default: 255.

Option "HWCursor" "*boolean***"**
Enable or disable the HW cursor. Default: on.

Option "MGASDRAM" "*boolean*"
Specify whether G100, G200 or G400 cards have SDRAM. The driver attempts to auto-detect this based on the card's PCI subsystem ID. This option may be used to override that auto-detection. The mga driver is not able to auto-detect the presence of of SDRAM on secondary heads in multihead configurations so this option will often need to be specified in multihead configurations. Default: auto-detected.

Option "NoAccel" "*boolean*"
Disable or enable acceleration. Default: acceleration is enabled.

Option "NoHal" "*boolean*"
Disable or enable loading the "mga_hal" module. Default: the module is loaded when available and when using hardware that it supports.

Option "OverclockMem"
Set clocks to values used by some commercial X-Servers (G100, G200 and G400 only). Default: off.

Option "Overlay" "*value*"
Enable 8+24 overlay mode. Only appropriate for depth 24. Recognized values are: "8,24", "24,8". Default: off. (**Note:** the G100 is unaccelerated in the 8+24 overlay mode due to a missing hardware feature.)

Option "PciRetry" "*boolean*"
Enable or disable PCI retries. Default: off.

Option "Rotate" "CW"

Option "Rotate" "CCW"
Rotate the display clockwise or counterclockwise. This mode is unaccelerated. Default: no rotation.

Option "ShadowFB" "*boolean*"
Enable or disable use of the shadow framebuffer layer. Default: off.

Option "SyncOnGreen" "*boolean*"
Enable or disable combining the sync signals with the green signal. Default: off.

Option "UseFBDev" "*boolean*"
Enable or disable use of on OS-specific fb interface (and is not supported on all OSs). See fbdevhw(4) for further information. Default: off.

Option "VideoKey" "*integer*"
This sets the default pixel value for the YUV video overlay key. Default: undefined.

Option "TexturedVideo" "*boolean*"
This has XvImage support use the texture engine rather than the video overlay. This option is only supported by the G200 and G400, and only in 16 and 32 bits per pixel. Default: off.

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7)

AUTHORS

Authors include: Radoslaw Kapitan, Mark Vojkovich, and also David Dawes, Guy Desbief, Dirk Hohndel, Doug Merritt, Andrew E. Mileski, Andrew van der Stock, Leonard N. Zubkoff, Andrew C. Aitchison.

NAME

neomagic − Neomagic video driver

SYNOPSIS

Section "Device"
Identifier "*devname***"**
Driver "neomagic"
...
EndSection

DESCRIPTION

neomagic is an XFree86 driver for the Neomagic graphics chipsets found in many laptop computers.

SUPPORTED HARDWARE

neomagic supports the following chipsets:

MagicGraph 128 (NM2070)

MagicGraph 128V (NM2090)

MagicGraph 128ZV (NM2093)

MagicGraph 128ZV+ (NM2097)

MagicGraph 128XD (NM2160)

MagicGraph 256AV (NM2200)

MagicGraph 256AV+ (NM2230)

MagicGraph 256ZX (NM2360)

MagicGraph 256XL+ (NM2380)

The driver supports depths 8, 15, 16 and 24 for all chipsets except the NM2070 which does not support depth 24. All depths are accelerated except for depth 24 which is only accelerated on NM2200 and newer models. All visuals are supported in depth 8. TrueColor and DirectColor visuals are supported in the other depths.

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details. This section only covers configuration details specific to this driver.

The following driver **Options** are supported

Option "NoAccel" "*boolean***"**
Disable or enable acceleration. Default: acceleration is enabled.

Option "SWCursor" "*boolean***"**
Disable or enable software cursor. Default: software cursor is disable and a hardware cursor is used.

Option "PCIBurst" "*boolean***"**
Disable or enable PCI burst modes. Default: enabled.

Option "Rotate" "CW"

Option "Rotate" "CCW"
Rotate the display clockwise or counterclockwise. This mode is unaccelerated. Default: no rotation.

Option "ShadowFB" "*boolean***"**
Enable or disable use of the shadow framebuffer layer. Default: off.

Option "OverlayMem" "*integer*"
Reserve the given amount of memory (in bytes) for the XVideo overlay. On boards with limited memory, display of large XVideo buffers might fail due to insufficient available memory. Using this option solves the problem at the expense of reducing the memory available for other operations. For full−resolution DVDs, 829440 bytes (720x576x2) are necessary.

Note
On some laptops using the 2160 chipset (MagicGraph 128XD) the following options are needed to avoid a lock-up of the graphic engine:
Option "XaaNoScanlineImageWriteRect"
Option "XaaNoScanlineCPUToScreenColorExpandFill"

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7)

AUTHORS

Authors include: Jens Owen, Kevin E. Martin, and also Egbert Eich, Mark Vojkovich, Alan Hourihane.

NAME
newport − Newport video driver

SYNOPSIS
Section "Device"
Identifier "*devname***"**
Driver "newport"
...
EndSection

DESCRIPTION
newport is an XFree86 driver for the SGI Indy’s and Indigo2’s newport video cards.

SUPPORTED HARDWARE
The **newport** driver supports the Newport (also called XL) cards found in SGI Indys and Indigo2s. It supports both the 8bit and 24bit versions of the Newport.

CONFIGURATION DETAILS
Please refer to XF86Config(5) for general configuration details. This section only covers configuration details specific to this driver.

The following driver options are supported:

Option "bitplanes" "*integer***"**
number of bitplanes of the board (8 or 24) Default: auto-detected.

Option "HWCursor" "*boolean***"**
Enable or disable the HW cursor. Default: on.

Option "BusID" "*integer***"**
Set to 1 for the Indigo2 XL. Default: 0

SEE ALSO
XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7)

AUTHORS
Authors: Guido GÜnther *agx@sigxcpu.org*

NAME

nsc – Nsc video driver

SYNOPSIS

Section "Device"
Identifier "*devname***"**
Driver "nsc"
...
EndSection

DESCRIPTION

nsc is an XFree86 driver for National Semiconductors GEODE processor family. It uses the DURANGO kit provided by National Semiconductor. The driver is accelerated, and provides support for the following framebuffer depths: 8, 16 and 24.

SUPPORTED HARDWARE

The **nsc** driver supports GXLV (5530 companion chip), SC1200, SC1400 and GX2 (5535 companion chip).

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details. This section only covers configuration details specific to this driver.

The driver will auto-detect the amount of video memory present for all chips. If the amount of memory is detected incorrectly, the actual amount of video memory should be specified with a **VideoRam** entry in the config file **"Device"** section.

The following driver **Options** are supported:

Option "SWCursor" "*boolean***"**
Enable or disable the SW cursor. Default: off.

Option "HWCursor" "*boolean***"**
Enable or disable the HW cursor. Default: on.

Option "NoAccel" "*boolean***"**
Disable or enable acceleration. Default: acceleration is enabled.

Option "NoCompression" "*boolean***"**
Disable or enable compression. Default: compression is enabled.

Option "ShadowFB" "*boolean***"**
Enable or disable use of the shadow framebuffer layer. Default: off.

Option "Rotate" "CW"
Rotate the display clockwise. This mode is unaccelerated, and uses the Shadow Frame Buffer layer. Default: no rotation.

Option "Rotate" "CCW"
Rotate the display counterclockwise. This mode is unaccelerated, and uses the Shadow Frame Buffer layer. Default: no rotation.

Option "FlatPanel" "*boolean***"**
This enables the FlatPanel display unit. The FlatPanel depends on the BIOS to do the Pnale h/w initialization. In GX2 based platforms with TFT part Flatpanel is enabled, and on CRT part is disabled. Default: off.

Option "OSMImageBuffers" "*integer***"**
This sets the number of scanline buffers to be allocated in offscreen memory for acceleration. This can take any value 0 will disable the allocation. Disabled if cannot allocate requested scanline memory. Default: 20.

Option "ColorKey" "*integer***"**
This sets the default pixel value for the YUV video overlay key. Default: 0.

The following **Options** are supported only on SC1200 based platforms:

Option "TV" "PAL-768x576"
Selects the PAL TV display mode 768x576 and the depth is forced to 16 bpp. Default: no TV.

Option "TV" "PAL-720x576"
Selects the PAL TV display mode 720x576 and the depth is forced to 16 bpp. Default: no TV.

Option "TV" "NTSC-720x480"
Selects the NTSC TV display mode 720x480 and the depth is forced to 16 bpp. Default: no TV.

Option "TV" "NTSC-640x480"
Selects the NTSC TV display mode 640x480 and the depth is forced to 16 bpp. Default: no TV.

Option "TV_Output" "COMPOSITE"
The selected TV mode output is coded for Composite signal. Default: no TV.

Option "TV_Output" "SVIDEO"
The selected TV mode output is coded for SVIDEO signal. Default: no TV.

Option "TV_Output" "YUV"
The selected TV mode output is coded for YUV signal. Default: no TV.

Option "TV_Output" "SCART"
The selected TV mode output is coded for SCART signal. Default: no TV.

Option "TVOverscan" x:yy:ww:hh"
This option will let only the viewable display area smaller to be able to view on TV. The parameters xx: X-offset, yy: Y-offset, ww: Viewable width, hh: Viewable height. Default: no TV.

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7)

AUTHOR

Author: Sarma V. Kolluru

NAME

nv – NVIDIA video driver

SYNOPSIS

Section "Device"
Identifier "*devname***"**
Driver "nv"
...
EndSection

DESCRIPTION

nv is an XFree86 driver for NVIDIA video cards. The driver supports 2D acceleration and provides support for the following framebuffer depths: 8, 15, 16 (except Riva128) and 24. All visual types are supported for depth 8, TrueColor and DirectColor visuals are supported for the other depths with the exception of the Riva128 which only supports TrueColor in the higher depths.

SUPPORTED HARDWARE

The **nv** driver supports PCI and AGP video cards based on the following NVIDIA chips:

RIVA 128 NV3

RIVA TNT NV4

RIVA TNT2 NV5

GeForce 256, QUADRO NV10

GeForce2, QUADRO2 NV11 & NV15

GeForce3, QUADRO DCC
NV20

nForce, nForce2 NV1A, NV1F

GeForce4, QUADRO4 NV17, NV18, NV25, NV28

GeForce FX, QUADRO FX
NV30, NV31, NV34, NV35, NV36, NV38

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details. This section only covers configuration details specific to this driver.

The driver auto-detects the chipset type and the amount of video memory present for all chips.

The following driver **Options** are supported:

Option "HWCursor" "*boolean***"**
Enable or disable the HW cursor. Default: on.

Option "NoAccel" "*boolean***"**
Disable or enable acceleration. Default: acceleration is enabled.

Option "UseFBDev" "*boolean***"**
Enable or disable use of on OS-specific fb interface (and is not supported on all OSs). See fbdevhw(4) for further information. Default: off.

Option "CrtcNumber" "*integer***"**
GeForce2 MX, nForce2, Quadro4, GeForce4, Quadro FX and GeForce FX may have two video outputs. The driver attempts to autodetect which one the monitor is connected to. In the case that autodetection picks the wrong one, this option may be used to force usage of a particular output. The options are "0" or "1". Default: autodetected.

Option "FlatPanel" "*boolean*"
The driver usually can autodetect the presence of a digital flat panel. In the case that this fails, this option can be used to force the driver to treat the attached device as a digital flat panel. With this driver, a digital flat panel will only work if it was POSTed by the BIOS, that is, the machine must have booted to the panel. If you have a dual head card you may also need to set the option CrtcNumber described above. Default: off.

Option "FPDither" "*boolean*"
Many digital flat panels (particularly ones on laptops) have only 6 bits per component color resolution. This option tells the driver to dither from 8 bits per component to 6 before the flat panel truncates it. This is only supported in depth 24 on GeForce2 MX, nForce2, GeForce4, Quadro4, Geforce FX and Quadro FX. Default: off.

Option "Rotate" "CW"

Option "Rotate" "CCW"
Rotate the display clockwise or counterclockwise. This mode is unaccelerated. Default: no rotation.

Note: The Resize and Rotate extension will be disabled if the Rotate option is used.

Option "ShadowFB" "*boolean*"
Enable or disable use of the shadow framebuffer layer. Default: off.

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7)

AUTHORS

Authors include: David McKay, Jarno Paananen, Chas Inman, Dave Schmenk, Mark Vojkovich

NAME

rendition − Rendition video driver

SYNOPSIS

Section "Device"
Identifier "*devname***"**
Driver "rendition"
...
EndSection

DESCRIPTION

rendition is an XFree86 driver for Rendition/Micron based video cards. The driver supports following framebuffer depths: 8, 15 (Verite V1000 only), 16 and 24. Acceleration and multi-head configurations are not supported yet, but are work in progress.

SUPPORTED HARDWARE

The **rendition** driver supports PCI and AGP video cards based on the following Rendition/Micron chips:

V1000 Verite V1000 based cards.

V2100 Verite V2100 based cards. Diamond Stealth II S220 is the only known such card.

V2200 Verite V2200 based cards.

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details. This section only covers configuration details specific to this driver.

The driver auto-detects the chipset type, but the following **ChipSet** names may optionally be specified in the config file **"Device"** section, and will override the auto-detection:

"v1000", "v2x00".

The driver will auto-detect the amount of video memory present for all chips. If the amount of memory is detected incorrectly, the actual amount of video memory should be specified with a **VideoRam** entry in the config file **"Device"** section.

The following driver **Options** are supported:

Option "SWCursor" "*boolean***"**
Disables use of the hardware cursor. Default: use HW-cursor.

Option "OverclockMem" "*boolean***"**
Increases the Mem/Sys clock to 125MHz/60MHz from standard 110MHz/50MHz. Default: Not overclocked.

Option "DacSpeed" "*MHz***"**
Run the memory at a higher clock. Useful on some cards with display glitches at higher resolutions. But adds the risk to damage the hardware. Use with caution.

Option "FramebufferWC" "*boolean***"**
If writecombine is disabled in BIOS, and you add this option in configuration file, then the driver will try to request writecombined access to the framebuffer. This can drastically increase the performance on unaccelerated server. Requires that "MTRR"-support is compiled into the OS-kernel. Default: Disabled for V1000, enabled for V2100/V2200.

Option "NoDDC" "*boolean***"**
Disable probing of DDC-information from your monitor. This information is not used yet and is only there for informational purposes. This might change before final XFree86 4.0 release. Safe to disable if you experience problems during startup of X-server. Default: Probe DDC.

Option "ShadowFB" "*boolean***"**
If this option is enabled, the driver will cause the CPU to do each drawing operation first into a shadow frame buffer in system virtual memory and then copy the result into video memory. If this option is not active, the CPU will draw directly into video memory. Enabling this option is

beneficial for those systems where reading from video memory is, on average, slower than the corresponding read/modify/write operation in system virtual memory. This is normally the case for PCI or AGP adapters, and, so, this option is enabled by default unless acceleration is enabled. Default: Enabled unless acceleration is used.

Option "Rotate" "CW"

Option "Rotate" "CCW"
Rotate the display clockwise or counterclockwise. This mode is unaccelerated. Default: no rotation.

Notes For the moment the driver defaults to not request write-combine for any chipset as there has been indications of problems with it. Use **Option "MTRR"** to let the driver request write-combining of memory access on the video board.

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7)

AUTHORS

Authors include: Marc Langenbach, Dejan Ilic

NAME

s3virge − S3 ViRGE video driver

SYNOPSIS

Section "Device"
Identifier "*devname***"**
Driver "s3virge"
...
[**Option** "optionname" ["optionvalue"]]
EndSection

DESCRIPTION

s3virge is an XFree86 driver for S3 based video cards. The driver is fully accelerated, and provides support for the following framebuffer depths: 8, 15, 16, and 24. All visual types are supported for depth 8, and TrueColor visuals are supported for the other depths. XVideo hardware up scaling is supported in depth 16 and 24 on the DX, GX, GX2, MX, MX+, and Trio3D/2X. Doublescan modes are supported and tested in depth 8 and 16 on DX, but disable XVideo. Doublescan modes on other chipsets are untested.

SUPPORTED HARDWARE

The **s3virge** driver supports PCI and AGP video cards based on the following S3 chips:

ViRGE 86C325

ViRGE VX 86C988

ViRGE DX 86C375

ViRGE GX 86C385

ViRGE GX2 86C357

ViRGE MX 86C260

ViRGE MX+
86C280

Trio 3D 86C365

Trio 3D/2X 86C362, 86C368

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details. This section only covers configuration details specific to this driver. All options names are case and white space insensitive when parsed by the server, for example, "virge vx" and "VIRGEvx" are equivalent.

The driver auto-detects the chipset type, but the following **ChipSet** names may optionally be specified in the config file **""Device""** section, and will override the auto-detection:

"virge", "86c325", "virge vx", "86c988", "virge dx", "86c375", "virge gx", "86c385", "virge gx2", "86c357", "virge mx", "86c260", "virge mx+", "86c280", "trio 3d", "86c365", "trio 3d/2x", "86c362", "86c368".

The following Cursor **Options** are supported:

Option "HWCursor" ["*boolean***"]**
Enable or disable the HW cursor. Default: on.

Option "SWCursor" ["*boolean***"]**
Inverse of "HWCursor". Default: off.

The following display **Options** are supported:

Option "ShadowFB" ["*boolean*"]
Use shadow framebuffer. Disables HW acceleration. Default: off.

Option "Rotate" "cw | ccw"
Rotate the screen CW - clockwise or CCW - counter clockwise. Disables HW Acceleration and HW Cursor, uses ShadowFB. Default: no rotation.

Option "XVideo" ["boolean**"]**
Disable XVideo support by using the off option. This changes FIFO settings which prevent screen noise for high-res modes. Default: on

The following video memory **Options** are supported:

Option "slow_edodram"
Switch the standard ViRGE to 2-cycle edo mode. Try this if you encounter pixel corruption on the ViRGE. Using this option will cause a large decrease in performance. Default: off.

Option "fpm_vram"
Switch the ViRGE/VX to fast page mode vram mode. Default: off.

Option "slow_dram | fast_dram"
Change Trio 3D and 3D/2X memory options. Default: Use BIOS defaults.

Option "early_ras_precharge | late_ras_precharge"
adjust memory parameters. One of these will us the same settings as your video card defaults, and using neither in the config file does the same. Default: none.

Option "set_mclk" "*integer***"**
sets the memory clock, where *integer* is in kHz, and *integer* <= 100000. Default: probe the memory clock value, and use it at server start.

Option "set_refclk" "*integer***"**
sets the ref clock for ViRGE MX, where *integer* is in kHz. Default: probe the memory clock value, and use it at server start.

The following acceleration and graphics engine **Options** are supported:

Option "NoAccel"
Disable acceleration. Very useful for determining if the driver has problems with drawing and acceleration routines. This is the first option to try if your server runs but you see graphic corruption on the screen. Using it decreases performance, as it uses software emulation for drawing operations the video driver can accelerate with hardware. Default: acceleration is enabled.

Option "UseFB"
There are two framebuffer rendering methods. fb and cfb. Both are available in the driver. fb is the newer and default method. To switch back to cfb use this option with no, off or other negative parameter. Default: on.

Option "fifo_aggressive | fifo_moderate | fifo_conservative"
alter the settings for the threshold at which the pixel FIFO takes over the internal memory bus to refill itself. The smaller this threshold, the better the acceleration performance of the card. You may try the fastest setting (**fifo_aggressive**) and move down if you encounter pixel corruption. The optimal setting will probably depend on dot-clock and on color depth. Note that specifying any of these options will also alter other memory settings which may increase performance, so trying **fifo_conservative** will in most cases be a slight benefit (this uses the chip defaults). If pixel corruption or transient streaking is observed during drawing operations then removing any fifo options is recommended. Default: none.

The following PCI bus **Options** are supported:

Option "pci_burst" ["*boolean*"]
will enable PCI burst mode. This should work on all but a few broken PCI chipsets, and will increase performance. Default: off.

Option "pci_retry" ["*boolean*"]
will allow the driver to rely on PCI Retry to program the ViRGE registers. **pci_burst** must be enabled for this to work. This will increase performance, especially for small fills/blits, because the driver does not have to poll the ViRGE before sending it commands to make sure it is ready. It should work on most recent PCI chipsets. Default: off.

The following ViRGE MX LCD **Options** are supported:

Option "lcd_center"

Option "set_lcdclk" "*integer*"
allows setting the clock for a ViRGE MX LCD display. *integer* is in Hz. Default: use probed value.

The following additional **Options** are supported:

Option "ShowCache" ["*boolean*"]
Enable or disable viewing offscreen cache memory. A development debug option. Default: off.

Option "mx_cr3a_fix" ["*boolean*"]
Enable or disable a cr3a fix added for ViRGE MX. Default: on.

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7)

KNOWN BUGS

The VideoRam generic driver parameter is presently ignored by the s3virge driver. On PPC this is reported to cause problems for 2M cards, because they may autodetect as 4M.

SUPPORT

For assistance with this driver, or XFree86 in general, check the XFree86 web site at http://www.xfree86.org. A FAQ is available on the web site at http://www.xfree86.org/FAQ/. If you find a problem with XFree86 or have a question not answered in the FAQ please use our bug report form available on the web site or send mail to XFree86@XFree86.org. When reporting problems with the driver send as much detail as possible, including chipset type, a server output log, and operating system specifics.

AUTHORS

Kevin Brosius, Matt Grossman, Harald Koenig, Sebastien Marineau, Mark Vojkovich.

NAME

savage − S3 Savage video driver

SYNOPSIS

Section "Device"
Identifier "*devname***"**
Driver "savage"
...
EndSection

DESCRIPTION

savage is an XFree86 driver for the S3 Savage family video accelerator chips. The **savage** driver supports PCI and AGP boards with the following chips:

Savage3D (8a20 and 8a21)

Savage4 (8a22)

Savage2000 (9102)

Savage/MX (8c10 and 8c11)

Savage/IX (8c12 and 8c13)

ProSavage PM133
(8a25)

ProSavage KM133
(8a26)

Twister (ProSavage PN133)
(8d01)

TwisterK (ProSavage KN133)
(8d02)

ProSavage DDR (8d03)

ProSavage DDR-K
(8d04)

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details. This section only covers configuration details specific to this driver.

The following driver **Options** are supported:

Option "HWCursor" "*boolean***"**

Option "SWCursor" "*boolean***"**
These two options interact to specify hardware or software cursor. If the SWCursor option is specified, any HWCursor setting is ignored. Thus, either "HWCursor off" or "SWCursor on" will force the use of the software cursor. On Savage/MX and Savage/IX chips which are connected to LCDs, a software cursor will be forced, because the Savage hardware cursor does not correctly track the automatic panel expansion feature. Default: hardware cursor.

Option "NoAccel" "*boolean***"**
Disable or enable acceleration. Default: acceleration is enabled.

Option "Rotate" "CW"

Option "Rotate" "CCW"
Rotate the desktop 90 degrees clockwise or counterclockwise. This option forces the ShadowFB option on, and disables acceleration. Default: no rotation.

Option "ShadowFB" "*boolean*"
Enable or disable use of the shadow framebuffer layer. This option disables acceleration. Default: off.

Option "LCDClock" "*frequency*"
Override the maximum dot clock. Some LCD panels produce incorrect results if they are driven at too fast of a frequency. If UseBIOS is on, the BIOS will usually restrict the clock to the correct range. If not, it might be necessary to override it here. The **frequency** parameter may be specified as an integer in Hz (135750000), or with standard suffixes like "k", "kHz", "M", or "MHz" (as in 135.75MHz).

Option "UseBIOS" "*boolean*"
Enable or disable use of the video BIOS to change modes. Ordinarily, the **savage** driver tries to use the video BIOS to do mode switches. This generally produces the best results with the mobile chips (/MX and /IX), since the BIOS knows how to handle the critical but unusual timing requirements of the various LCD panels supported by the chip. To do this, the driver searches through the BIOS mode list, looking for the mode which most closely matches the XF86Config mode line. Some purists find this scheme objectionable. If you would rather have the **savage** driver use your mode line timing exactly, turn off the UseBios option. Default: on (use the BIOS).

Option "ShadowStatus" "*boolean*"
Enables the use of a shadow status register. There is a chip bug in the Savage graphics engine that can cause a bus lock when reading the engine status register under heavy load, such as when scrolling text or dragging windows. The bug affects about 4% of all Savage users. If your system hangs regularly while scrolling text or dragging windows, try turning this option on. This uses an alternate method of reading the engine status which is slightly more expensive, but avoids the problem. Default: off (use normal status register).

FILES

savage_drv.o

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7)

AUTHORS

Authors include Tim Roberts (timr@probo.com) and Ani Joshi (ajoshi@unixbox.com) for the 4.0 version, and Tim Roberts and S. Marineau for the 3.3 driver from which this was derived.

NAME

siliconmotion − Silicon Motion video driver

SYNOPSIS

Section "Device"
  Identifier "*devname***"**
  Driver "siliconmotion"
  ...
  [**Option** "optionname" ["optionvalue"]]
EndSection

DESCRIPTION

siliconmotion is an XFree86 driver for Silicon Motion based video cards. The driver is fully accelerated, and provides support for the following framebuffer depths: 8, 16, and 24. All visual types are supported for depth 8, and TrueColor visuals are supported for the other depths.

SUPPORTED HARDWARE

The **siliconmotion** driver supports PCI and AGP video cards based on the following Silicon Motion chips:

Lynx SM910

LynxE SM810

Lynx3D SM820

LynxEM SM710

LynxEM+ SM712

Lynx3DM SM720

Cougar3DR SM730

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details. This section only covers configuration details specific to this driver. All options names are case and white space insensitive when parsed by the server, for example, "lynxe" and "LynxE" are equivalent.

The driver auto-detects the chipset type, but the following **ChipSet** names may optionally be specified in the config file **"Device"** section, and will override the auto-detection:

"lynx", "lynxe", "lynx3d", "lynxem", "lynxem+", "lynx3dm", "cougar3dr".

The following Cursor **Options** are supported:

Option "HWCursor" "*boolean***"**
Enable or disable the HW cursor. Default: on.

Option "SWCursor" "*boolean***"**
Inverse of "HWCursor". Default: off.

The following display **Options** are supported:

Option "ShadowFB" "*boolean***"**
Use shadow framebuffer. Default: off.

Option "Rotate" "CW"

Option "Rotate" "CCW"
Rotate the screen CW - clockwise or CCW - counter clockwise. Uses ShadowFB. Default: no rotation.

Option "VideoKey" "*integer***"**
Set the video color key. Default: a little off full blue.

Option "ByteSwap" "*boolean*"
Turn on byte swapping for capturing using SMI demo board. Default: off.

Option "Interlaced" "*boolean*"
Turn on interlaced video capturing. Default: off.

Option "UseBIOS" "*boolean*"
Use the BIOS to set the modes. This is used for custom panel timings. Default: on.

Option "ZoomOnLCD" "*boolean*"
Allow changing resolution on LCD (Ctrl-Alt-Plus and Ctrl-Alt-Minus). Default: off.

The following video memory **Options** are supported:

Option "set_mclk" "*integer*"
sets the memory clock, where *integer* is in kHz, and *integer* <= 100000. Default: probe the memory clock value, and use it at server start.

The following acceleration and graphics engine **Options** are supported:

Option "NoAccel"
Disable acceleration. Very useful for determining if the driver has problems with drawing and acceleration routines. This is the first option to try if your server runs but you see graphic corruption on the screen. Using it decreases performance, as it uses software emulation for drawing operations the video driver can accelerate with hardware. Default: acceleration is enabled.

Option "fifo_aggressive"

Option "fifo_moderate"

Option "fifo_conservative"
alter the settings for the threshold at which the pixel FIFO takes over the internal memory bus to refill itself. The smaller this threshold, the better the acceleration performance of the card. You may try the fastest setting (**fifo_aggressive**) and move down if you encounter pixel corruption. The optimal setting will probably depend on dot-clock and on color depth. Note that specifying any of these options will also alter other memory settings which may increase performance, so trying **fifo_conservative** will in most cases be a slight benefit (this uses the chip defaults). If pixel corruption or transient streaking is observed during drawing operations then removing any fifo options is recommended. Default: none.

The following PCI bus **Options** are supported:

Option "pci_burst" "*boolean*"
will enable PCI burst mode. This should work on all but a few broken PCI chipsets, and will increase performance. Default: off.

Option "pci_retry" "*boolean*"
will allow the driver to rely on PCI Retry to program the ViRGE registers. **pci_burst** must be enabled for this to work. This will increase performance, especially for small fills/blits, because the driver does not have to poll the ViRGE before sending it commands to make sure it is ready. It should work on most recent PCI chipsets. Default: off.

The following additional **Options** are supported:

Option "ShowCache" "*boolean*"
Enable or disable viewing offscreen cache memory. A development debug option. Default: off.

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7)

SUPPORT

For assistance with this driver, or XFree86 in general, check the XFree86 web site at http://www.xfree86.org. A FAQ is available on the web site at http://www.xfree86.org/FAQ/. If you find a problem with XFree86 or have a question not answered in the FAQ please use our bug report form available on the web site or send mail to XFree86@XFree86.org. When reporting problems with the driver send as much detail as possible, including chipset type, a server output log, and operating system specifics.

AUTHORS

Kevin Brosius, Matt Grossman, Harald Koenig, Sebastien Marineau, Mark Vojkovich, Frido Garritsen, Corvin Zahn.

NAME

sis − SiS video driver

SYNOPSIS

Section "Device"
Identifier "*devname***"**
Driver "sis"
...
EndSection

DESCRIPTION

sis is an XFree86 driver for SiS (Silicon Integrated Systems) video chips. The driver is accelerated, and provides support for colordepths of 8, 16 and 24 bpp. XVideo, Render and other extensions are supported as well.

SUPPORTED HARDWARE

The **sis** driver supports PCI and AGP video cards based on the following chipsets:

SiS5597/5598 SiS530/620 SiS6326/AGP/DVD SiS300/305 SiS540 SiS630/730 SiS315/H/PRO SiS550/551/552 SiS650/651/M650/661FX/M661FX/M661MX/740/741 SiS330 (Xabre) SiS760

In the following text, the following terms are used:

old series for SiS5597/5598, 530/620 and 6326/AGP/DVD

300 series for SiS300/305, 540 and 630/730

315/330 series for SiS315/H/PRO, 55x and (M)65x/(M)661xX/74x, 330, 760

Note: Support for 661FX, M661FX, M661MX, 741 and 760 is untested.

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details. This section only covers configuration details specific to this driver.

Detailed information on all supported options can be obtained at http://www.winischhofer.net/linux-sisvga.shtml

This manpage only covers a subset of the supported options.

1. For all supported chipsets

The following driver **Options** are supported on all chipsets:

Option "NoAccel" "*boolean***"**
Disable or enable 2D acceleration. Default: acceleration is enabled.

Option "HWCursor" "*boolean***"**
Enable or disable the HW cursor. Default: HWCursor is on.

Option "SWCursor" "*boolean***"**
The opposite of HWCursor. Default: SWCursor is off.

Option "Rotate" "CW"
Rotate the display clockwise. This mode is unaccelerated, and uses the Shadow Frame Buffer layer. Using this option disables the Resize and Rotate extension (RandR). Default: no rotation.

Option "Rotate" "CCW"
Rotate the display counterclockwise. This mode is unaccelerated, and uses the Shadow Frame Buffer layer. Using this option disables the Resize and Rotate extension (RandR). Default: no rotation.

Option "ShadowFB" "*boolean***"**
Enable or disable use of the shadow framebuffer layer. Default: Shadow framebuffer is off.

Option "CRT1Gamma" "*boolean*"
Enable or disable gamma correction. Default: Gamma correction is on.

2. Old series specific information

The driver will auto-detect the amount of video memory present for all these chips, but in the case of the 6326, it will limit the memory size to 4MB. This is because the 6326's 2D engine can only address 4MB. The remaining memory seems to be intended for 3D texture data, since only the 3D engine can address RAM above 4MB. However, you can override this limitation using the **"VideoRAM"** option in the Device section if your board has more than 4MB and you need to use it. However, 2D acceleration, Xvideo and the HWCursor will be disabled in this case.

The driver will also auto-detect the maximum dotclock and DAC speed. If you have problems getting high resolutions because of dot clock limitations, try using the **"DacSpeed"** option, also in the Device section. However, this is not recommended for the 6326. For this chip, the driver has two built-in modes for high resolutions which you should use instead. These are named **"SIS1280x1024-75"** and **"SIS1600x1200-60"** and they will be added to the list of default modes. To use these modes, just place them in your Screen section. Example:

Modes "SIS1600x1200-60" "SIS1280x1024x75" "1024x768" ...

Of these modes, 1280x1024 is only available at 8, 15 and 16bpp. 1600x1200 is available at 8bpp only.

TV support for the 6326

TV output is supported for the 6326. The driver will auto detect a TV connected and in this case add the following modes to the list of default modes: "PAL800x600", "PAL800x600U", "PAL720x540", "PAL640x480", "NTSC640x480", "NTSC640x480U" and "NTSC640x400". Use these modes like the hires modes described above.

The following driver **Options** are supported on the old series:

Option "TurboQueue" "*boolean*"
Enable or disable TurboQueue mode. Default: off for SIS530/620, on for the others

Option "FastVram" "*boolean*"
Enable or disable FastVram mode. Enabling this sets the video RAM timing to one cycle per read operation instead of two cycles. Disabling this will set two cycles for read and write operations. Leaving this option out uses the default, which varies depending on the chipset.

Option "NoHostBus" "*boolean*"
(SiS5597/5598 only). Disable CPU-to-VGA host bus support. This speeds up CPU to video RAM transfers. Default: Host bus is enabled.

Option "NoXVideo" "*boolean*"
Disable XV (XVideo) extension support. Default: XVideo is on.

Option "NoYV12" "*boolean*"
Disable YV12 Xv support. This might me required due to hardware bugs in some chipsets. Disabling YV12 support forces Xv-aware applications to use YUV2 or XShm for video output. Default: YV12 support is on.

Option "TVStandard" "*string*"
(6326 only) Valid parameters are **PAL** or **NTSC.** The default is set by a jumper on the card.

Option "TVXPosOffset" "*integer*"
(6326 only) This option allows tuning the horizontal position of the image for TV output. The range is from -16 to 16. Default: 0

Option "TVYPosOffset" "*integer*"
(6326 only) This option allows tuning the vertical position of the image for TV output. The range is from -16 to 16. Default: 0

Option "SIS6326TVEnableYFilter" "*boolean*"
(6326 only) This option allows enabling/disabling the Y (chroma) filter for TV output.

Option "SIS6326TVAntiFlicker" "*string*"
(6326 only) This option allow enabling/disabling the anti flicker facility for TV output. Possible parameters are **OFF, LOW, MED, HIGH** or **ADAPTIVE.** By experience, **ADAPTIVE** yields the best results, hence it is the default.

2. 300 and 315/330 series specific information

The 300 and 315/330 series feature two CRT controllers and very often come with a video bridge for controlling LCD and TV output. Hereinafter, the term **CRT1** refers to the VGA output of the chip, and **CRT2** refers to either LCD, TV or secondary VGA. Due to timing reasons, only one CRT2 output can be active at the same time. But this limitation does not apply to using CRT1 and CRT2 at the same time which makes it possible to run the driver in dual head mode.

The driver supports the following video bridges:

SiS301 SiS301B(-DH) SiS301C SiS301LV(X) SiS302LV

Instead of a video bridge, some machines have a **LVDS** transmitter to control LCD panels, and a **Chrontel 7005** or **7019** for TV output. All these are supported as well.

About TV output

On the SiS301 and the Chrontel 7005, only resolutions up to 800x600 are supported. On all others, resolutions up to 1024x768 are supported. However, due to a hardware bug, Xvideo might be distorted on SiS video bridges if running NTSC or PAL-M at 1024x768.

About XVideo support

XVideo is supported on all chipsets of both families. However, there are some differences in hardware features which cause limitations. The 300 series as well as the SiS55x, M650, 651, 661FX, M661FX, and 741 support two video overlays. The SiS315/H/PRO, 650/740 and 330 support only one such overlay. On chips with two overlays, one overlay is used for CRT1, the other for CRT2. On the other chipsets, the option **"XvOnCRT2"** can be used to select the desired output channel.

About Merged Framebuffer support

This mode is strongly recommended over Xinerama. Please see http://www.winischhofer.net/linux-sisvga.shtml for detailed information.

About dual-head support

Dual head mode has some limitations as regards color depth and resolution. Due to memory bandwidth limits, CRT1 might have a reduced maximum refresh rate if running on higher resolutions than 1280x1024.

Colordepth 8 is not supported when running in dual head mode.

The following driver **Options** are supported on the 300 and 315/330 series:

Option "NoXVideo" "*boolean*"
Disable XV (XVideo) extension support. Default: XVideo is on.

Option "XvOnCRT2" "*boolean*"
On chipsets with only one video overlay, this option can used to bind the overlay to CRT1 (if a monitor is detected and if this option is either unset or set to **false**) or CRT2 (if a CRT2 device is detected or forced, and if this option is set to **true**). If either only CRT1 or CRT2 is detected, the driver decides automatically. In Merged Framebuffer mode, this option is ignored. Default: overlay is used on CRT1

Option "ForceCRT1" "*boolean*"
Force CRT1 to be on of off. If a monitor is connected, it will be detected during server start. However, some old monitors are not detected correctly. In such cases, you may set this option to **on** in order to make the driver initialize CRT1 anyway. If this option is set to **off** , the driver will switch off CRT1. Default: auto-detect

Option "ForceCRT2Type" "*string*"
Force display type to one of: **NONE** , **TV** , **SVIDEO** , **COMPOSITE** , **SVIDEO+COMPOSITE** , **SCART** , **LCD** , **VGA** ; **NONE** will disable CRT2. The SVIDEO, COMPOSITE, SVIDEO+COMPOSITE and SCART parameters are for SiS video bridges only and can be used to force the driver to use a specific TV output connector (if present). Default: auto detect.

Option "CRT2Gamma" "*boolean*"
Enable or disable gamma correction for CRT2. Only supported for SiS video bridges. Default: Gamma correction for CRT2 is on.

Option "TVStandard" "*string*"
Force the TV standard to either **PAL** or **NTSC.** On some machines with 630, 730 and the 315/330 series, **PALM** , **PALN** and **NTSCJ** are supported as well. Default: BIOS setting.

Option "TVXPosOffset" "*integer*"
This option allows tuning the horizontal position of the image for TV output. The range is from -32 to 32. Not supported on the Chrontel 7019. Default: 0

Option "TVYPosOffset" "*integer*"
This option allows tuning the vertical position of the image for TV output. The range is from -32 to 32. Not supported on the Chrontel 7019. Default: 0

Option "SISTVXScale" "*integer*"
This option selects the horizontal zooming level for TV output. The range is from -16 to 16. Only supported on SiS video bridges. Default: 0

Option "SISTVYScale" "*integer*"
This option selects the vertical zooming level for TV output in the following modes: 640x480, 800x600. On the 315/330 series, also 720x480, 720x576 and 768x576. The range is from -4 to 3. Only supported on SiS video bridges. Default: 0

Option "CHTVOverscan" "*boolean*"
On machines with a Chrontel TV encoder, this can be used to force the TV mode to overscan or underscan. **on** means overscan, **off** means underscan. Default: BIOS setting.

Option "CHTVSuperOverscan" "*boolean*"
On machines with a Chrontel 7005 TV encoder, this option enables a super-overscan mode. This is only supported if the TV standard is PAL. Super overscan will produce an image on the TV which is larger than the viewable area.

The driver supports many more options. Please see http://www.winischhofer.net/linuxsisvga.shtml for more information.

3. 300 series specific information

DRI is supported on the 300 series only. On Linux, DRI requires the kernel's SiS framebuffer driver (**sisfb**) and some other modules which come with either the kernel or XFree86.

Sisfb takes care of memory management for texture data. In order to prevent the X Server and sisfb from overwriting each other's data, sisfb reserves an amount of video memory for the X driver. This amount can either be selected using sisfb's mem parameter, or auto-selected depending on the amount of total video RAM available.

Sisfb can be used for memory management only, or as a complete framebuffer driver. If you start sisfb with a valid mode (ie you gain a graphical console), the X driver can communicate with sisfb and doesn't require any manual configuration for finding out about the video memory it is allowed to use. However, if you are running a 2.4 series Linux kernel and use sisfb for video memory management only, ie you started sisfb with mode=none and still have a text mode console, there is no communication between sisfb and the X driver. For this purpose, the

Option "MaxXFBMem" "*integer*"

exists. This option must be set to the same value as given to sisfb through its "mem" parameter, ie the amount of memory to use for X in kilobytes.

If you started sisfb without the mem argument, sisfb will reserve

12288KB if more than 16MB of total video RAM is available,

8192KB if between 12 and 16MB of video RAM is available,

4096KB in all other cases.

If you intend to use DRI, I recommend setting the total video memory in the BIOS to 64MB in order to at least overcome the lack of memory swap functions.

Option "DRI" "*boolean*"
This option allows enabling or disabling DRI. By default, DRI is on.

Option "AGPSize" "*integer*"
This option allows selecting the amount of AGP memory to be used for DRI. The amount is to be specified in megabyte, the default is 8.

KNOWN BUGS

none.

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7)

http://www.winischhofer.net/linuxsisvga.shtml for more information and updates

AUTHORS

Authors include: Alan Hourihane, Mike Chapman, Juanjo Santamarta, Mitani Hiroshi, David Thomas, Sung-Ching Lin, Ademar Reis, Thomas Winischhofer

NAME
sunbw2 − BW2 video driver

SYNOPSIS
Section "Device"
Identifier "*devname***"**
Driver "sunbw2"
...
EndSection

DESCRIPTION
sunbw2 is an XFree86 driver for Sun BW2 video cards. THIS MAN PAGE NEEDS TO BE FILLED IN.

SUPPORTED HARDWARE
The **sunbw2** driver supports...

CONFIGURATION DETAILS
Please refer to XF86Config(5) for general configuration details. This section only covers configuration details specific to this driver.

SEE ALSO
XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7)

AUTHORS
Authors include: Jakub Jelinek <jakub@redhat.com>

NAME

suncg14 − CG14 video driver

SYNOPSIS

Section "Device"
Identifier "*devname***"**
Driver "suncg14"
...
EndSection

DESCRIPTION

suncg14 is an XFree86 driver for Sun CG14 video cards. THIS MAN PAGE NEEDS TO BE FILLED IN.

SUPPORTED HARDWARE

The **suncg14** driver supports...

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details. This section only covers configuration details specific to this driver.

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7)

AUTHORS

Authors include: Jakub Jelinek <jakub@redhat.com>

NAME

suncg3 – CG3 video driver

SYNOPSIS

Section "Device"
Identifier "*devname***"**
Driver "suncg3"
...
EndSection

DESCRIPTION

suncg3 is an XFree86 driver for Sun CG3 video cards. THIS MAN PAGE NEEDS TO BE FILLED IN.

SUPPORTED HARDWARE

The **suncg3** driver supports...

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details. This section only covers configuration details specific to this driver.

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7)

AUTHORS

Authors include: Jakub Jelinek <jakub@redhat.com>

NAME

suncg6 – GX/Turbo GX video driver

SYNOPSIS

Section "Device"
Identifier "*devname***"**
Driver "suncg6"
...
EndSection

DESCRIPTION

suncg6 is an XFree86 driver for Sun GX and Turbo GX (also known as cgsix) video cards. THIS MAN PAGE NEEDS TO BE FILLED IN.

SUPPORTED HARDWARE

The **suncg6** driver supports...

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details. This section only covers configuration details specific to this driver.

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7)

AUTHORS

Authors include: Jakub Jelinek <jakub@redhat.com>

NAME

ffb − SUNFFB video driver

SYNOPSIS

Section "Device"
 Identifier "*devname*"
 Driver "sunffb"
 ...
EndSection

DESCRIPTION

ffb is an XFree86 driver for Sun Creator, Creator 3D and Elite 3D video cards. THIS MAN PAGE NEEDS TO BE FILLED IN.

SUPPORTED HARDWARE

The **ffb** driver supports...

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details. This section only covers configuration details specific to this driver.

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7)

AUTHORS

Authors include: Jakub Jelinek <jakub@redhat.com>, David S. Miller <davem@redhat.com>, Michal Rehacek <majkl@iname.com>

NAME

sunleo – Leo video driver

SYNOPSIS

Section "Device"
  Identifier "*devname***"**
  Driver "sunleo"
  ...
EndSection

Section "Screen"
  ...
  Device "*devname***"**
  ...
  DefaultDepth 32
  ...
EndSection

DESCRIPTION

leo is an XFree86 driver for Sun Leo (ZX) video cards.

Also known as the ZX or T(urbo)ZX, Leo is a 24 bit accelerated 3D graphics card. Both cards are double-width, but the TZX also requires extra cooling in the form of an additional double-width fan card, so effectively takes up 4 SBus slots.

SUPPORTED HARDWARE

The **leo** driver supports all Sun stations that include a Leo chipset:

Workstations:

Sun 4/15, 4/30, 4/75
SPARCstation 5, 10, 20
Ultra 1, 1E, 2

Servers:

SPARCserver 1000,
SPARCcenter 2000

CONFIGURATION DETAILS

You must set "DefaultDepth" to "32" in the Screen Section.
Please refer to XF86Config(5) for general configuration details. This section only covers configuration details specific to this driver.

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7)

AUTHORS

Driver authors include: Jakub Jelinek <jakub@redhat.com>
Man page: Arnaud Quette <arnaud.quette@mgeups.com>

NAME

suntcx − TCX video driver

SYNOPSIS

Section "Device"
Identifier "*devname***"**
Driver "suntcx"
...
EndSection

DESCRIPTION

suntcx is an XFree86 driver for Sun TCX video cards. THIS MAN PAGE NEEDS TO BE FILLED IN.

SUPPORTED HARDWARE

The **suntcx** driver supports...

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details. This section only covers configuration details specific to this driver.

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7)

AUTHORS

Authors include: Jakub Jelinek <jakub@redhat.com>

NAME

tdfx − 3Dfx video driver

SYNOPSIS

Section "Device"
Identifier "*devname***"**
Driver "tdfx"
...
EndSection

DESCRIPTION

tdfx is an XFree86 driver for 3Dfx video cards.

SUPPORTED HARDWARE

The **tdfx** driver supports Voodoo Banshee, Voodoo3, Voodoo4 and Voodoo5 cards.

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details. This section only covers configuration details specific to this driver.

The following driver **Options** are supported:

Option "NoAccel" "*boolean***"**
Disable or enable acceleration. Default: acceleration is enabled.

Option "SWCursor" "*boolean***"**
Disable or enable software cursor. Default: software cursor is disable and a hardware cursor is used for configurations where the hardware cursor is available.

Option "DRI" "*boolean***"**
Disable or enable DRI support. By default, DRI is on.

Option "TexturedVideo" "*boolean***"**
This has XvImage support use the texture engine rather than the video overlay.

Option "VideoKey" "*integer***"**
This sets the default pixel value for the YUV video overlay key. Default: undefined.

The following additional **Options** are supported:

Option "ShowCache" "*boolean***"**
Enable or disable viewing offscreen cache memory. A development debug option. Default: off.

FILES

tdfx_drv.o

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7)

AUTHORS

Authors include: ...

NAME

trident − Trident video driver

SYNOPSIS

Section "Device"
Identifier "*devname***"**
Driver "trident"
...
EndSection

DESCRIPTION

trident is an XFree86 driver for Trident video cards. The driver is accelerated, and provides support for the following framebuffer depths: 1, 4, 8, 15, 16, and 24. Multi-head configurations are supported. The XvImage extension is supported on TGUI96xx and greater cards.

SUPPORTED HARDWARE

The **trident** driver supports PCI,AGP and ISA video cards based on the following Trident chips:

Blade Blade3D, CyberBlade series i1, i7 (DSTN), i1, i1 (DSTN), Ai1, Ai1 (DSTN), CyberBlade/e4, CyberBladeXP, CyberBladeAi1/XP, BladeXP

Image 3DImage975, 3DImage985, Cyber9520, Cyber9525, Cyber9397, Cyber9397DVD

ProVidia 9682, 9685, Cyber9382, Cyber9385, Cyber9388

TGUI 9440AGi, 9660, 9680

ISA/VLBus 8900C, 8900D, 9000, 9200CXr, Cyber9320, 9400CXi, 9440AGi These cards have been ported but need further testing and may not work.

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details. This section only covers configuration details specific to this driver.

The following driver **Options** are supported:

Option "SWCursor" "*boolean***"**
Enable or disable the SW cursor. Default: off.

Option "NoAccel" "*boolean***"**
Disable or enable acceleration. Default: acceleration is enabled.

Option "PciRetry" "*boolean***"**
Enable or disable PCI retries. Default: off.

Option "CyberShadow" "*boolean***"**
For Cyber chipsets only, turn off shadow registers. If you only see a partial display - this may be the option for you. Default: on.

Option "CyberStretch" "*boolean***"**
For Cyber chipsets only, turn on stretching. When the resolution is lower than the LCD's screen, this option will stretch the graphics mode to fill the entire LCD. Default: off.

Option "ShadowFB" "*boolean***"**
Enable or disable use of the shadow framebuffer layer. Default: off.

Option "VideoKey" "*integer***"**
This sets the default pixel value for the YUV video overlay key. Default: undefined.

Option "TVChipset" "*string***"**
This sets the TV chipset. Options are CH7005 or VT1621. Default: off.

Option "TVSignal" "*integer***"**
This sets the TV signalling. Options are 0 for NTSC or 1 for PAL. Default: undefined.

Option "NoPciBurst" "*boolean*"
Turn off PCI burst mode, PCI Bursting is on by default. Default: off.

Option "XvHsync" "*integer*"
Override the default Horizontal-sync value for the Xv extension. This is used to center the Xv image on the screen. By default the values are assigned based on the video card. Default: 0.

Option "XvVsync" "*integer*"
Override the default Vertical-sync value for the Xv extension. This is used to center the Xv image on the screen. By default the values are assigned based on the video card. Default: 0.

Option "XvBskew" "*integer*"
Override the default Bottom skew value for the Xv extension. This is used to extend the Xv image on the screen at the bottom. By default the values are assigned based on the video card. Default: 0.

Option "XvRskew" "*integer*"
Override the default Right skew value for the Xv extension. This is used to extend the Xv image on the screen at the right. By default the values are assigned based on the video card. Default: 0.

Option "Display" "*string*"
Override the display. Possible values are "CRT", "LCD" and "Dual". Please note that this option is only experimentally. Default: Use display active when X started.

Option "Display1400" "*boolean*"
Inform driver to expect 1400x1050 display instead of a 1280x1024. Default: off.

Option "GammaBrightness" "*string*"
Set display gamma value and brightness. "*string*" is "*gamma*, *brightness*", where *gamma* is a floating point value greater than 0 and less or equal to 10. *brightness* is an integer value greater or equal to 0 and less than 128. Default: gamma and brightness control is turned off. Note: This is not supported on all chipsets.

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7)

AUTHOR

Author: Alan Hourihane

NAME

tseng − Tseng Labs video driver

SYNOPSIS

Section "Device"
Identifier "*devname***"**
Driver "tseng"
...
EndSection

DESCRIPTION

tseng is an XFree86 driver for Tseng Labs video cards. THIS MAN PAGE NEEDS TO BE FILLED IN.

SUPPORTED HARDWARE

The **tseng** driver supports...

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details. This section only covers configuration details specific to this driver.

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7)

AUTHORS

Authors include: ...

NAME

v4l − video4linux driver

SYNOPSIS

Section "Module"
...
Load "v4l"
EndSection

DESCRIPTION

v4l is an XFree86 driver for video4linux cards. It provides a Xvideo extention port for video overlay. Just add the driver to the module list within the module section of your XF86Config file if you want to use it. There are no config options.

Note that the the extmod module is also required for the Xvideo support (and lots of other extentions too).

SUPPORTED HARDWARE

The **v4l** driver works with every piece of hardware which is supported by a video4linux (kernel-) device driver and is able to handle video overlay.

bt848/bt878-based TV cards are the most popular hardware these days.

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details. This section only covers configuration details specific to this driver.

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7)

AUTHORS

Authors include: Gerd Knorr <kraxel@bytesex.org>

NAME

vesa – Generic VESA video driver

SYNOPSIS

Section "Device"
  Identifier "*devname***"**
  Driver "vesa"
  ...
EndSection

DESCRIPTION

vesa is an XFree86 driver for generic VESA video cards. It can drive most VESA-compatible video cards, but only makes use of the basic standard VESA core that is common to these cards. The driver supports depths 8, 15 16 and 24.

SUPPORTED HARDWARE

The **vesa** driver supports most VESA-compatible video cards. There are some known exceptions, and those should be listed here.

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details. This section only covers configuration details specific to this driver.

The driver auto-detects the presence of VESA-compatible hardware. The **ChipSet** name may optionally be specified in the config file **"Device"** section, and will override the auto-detection:

"vesa"

The following driver **Options** are supported:

Option "ShadowFB" "*boolean***"**
Enable or disable use of the shadow framebuffer layer. Default: on.

This option is recommended for performance reasons.

SEE ALSO

XFree86(1), XF86Config(5), xf86cfg(1), xf86config(1), Xserver(1), X(7)

AUTHORS

Authors include: Paulo César Pereira de Andrade.

NAME

vga – Generic VGA video driver

SYNOPSIS

Section "Device"
Identifier "*devname***"**
Driver "vga"
...
EndSection

DESCRIPTION

vga is an XFree86 driver for generic VGA video cards. It can drive most VGA-compatible video cards, but only makes use of the basic standard VGA core that is common to these cards. The driver supports depths 1, 4 and 8. All relevant visual types are supported at each depth. Multi-head configurations are supported in combination with some other drivers, but only when the **vga** driver is driving the primary head.

SUPPORTED HARDWARE

The **vga** driver supports most VGA-compatible video cards. There are some known exceptions, and those should be listed here.

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details. This section only covers configuration details specific to this driver.

The driver auto-detects the presence of VGA-compatible hardware. The **ChipSet** name may optionally be specified in the config file **"Device"** section, and will override the auto-detection:

"generic"

The driver will only use 64k of video memory for depth 1 and depth 8 operation, and 256k of video memory for depth 4 (this is the standard VGA limit).

When operating at depth 8, only a single built-in 320x200 video mode is available. At other depths there is more flexibility regarding mode choice.

The following driver **Options** are supported:

Option "ShadowFB" "*boolean***"**

Enable or disable use of the shadow framebuffer layer. Default: off.

This option is recommended for performance reasons when running at depths 1 and 4, especially when using modern PCI-based hardware. It is required when using those depths in a multi-head configuration where one or more of the other screens is operating at a different depth.

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7)

AUTHORS

Authors include: Marc La France, David Dawes, and Dirk Hohndel.

NAME

vmware − VMware SVGA video driver

SYNOPSIS

Section "Device"
Identifier "*devname***"**
Driver "vmware"
...
EndSection

DESCRIPTION

vmware is an XFree86 driver for VMware virtual video cards.

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details. This section only covers configuration details specific to this driver.

The driver auto-detects the version of any virtual VMware SVGA adapter.

The following driver **Options** are supported:

Option "HWCursor" "*boolean***"**
Enable or disable the HW cursor. Default: off.

Option "NoAccel" "*boolean***"**
Disable or enable acceleration. Default: acceleration is enabled.

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7)

AUTHORS

Copyright (c) 1999-2001 VMware, Inc.

NAME

gtf - calculate VESA GTF mode lines

SYNOPSIS

gtf *h-resolution v-resolution refresh* [**−v|−−verbose**] [**−f|−−fbmode**] [**−x|−−xf86mode**]

DESCRIPTION

Gtf is a utility for calculating VESA GTF modes. Given the desired horizontal and vertical resolutions and refresh rate (in Hz), the parameters for a matching VESA GTF mode are printed out. Two output formats are supported: mode lines suitable for the XFree86 **XF86Config(5)** file, and mode parameters suitable for the Linux **fbset(8)** utility.

OPTIONS

−v|−−verbose
: Enable verbose printouts This shows a trace for each step of the computation.

−x|−−xf86mode
: Print the mode parameters as XFree86-style mode lines. This is the default format.

−f|−−fbset
: Print the mode parameters in a format suitable for **fbset(8)**.

SEE ALSO

XF86Config(5)

AUTHOR

Andy Ritger.

This program is based on the Generalized Timing Formula (GTF(TM)) Standard Version: 1.0, Revsion: 1.0. The GTF Excel(TM) spreadsheet, a sample (and the definitive) implementation of the GTF Timing Standard is available at <ftp://ftp.vesa.org/pub/GTF/VTF_V1R1.xls>.

NAME

kbd_mode − recover the PC console keyboard

SYNOPSIS

kbd_mode [-a -u]

DESCRIPTION

Kbd_mode resets the PC console keyboard to a rational state.

OPTIONS

The following options are supported:

−a Set the keyboard so that ASCII characters are read from the console.

−u Set the keyboard so that undecoded keyboard values are read from the console.

EXAMPLES

If the server crashes or otherwise fails to put the keyboard back in ascii mode when it exits, it can leave your keyboard dead. If you are able to login remotely, you can reset it typing:

```
kbd_mode -a
```

Conversely, changing the keyboard to ascii mode while the server is running will make the keyboard appear to be dead while the the mouse continues to work. Again, if you are able to login remotely, you can reset it typing:

```
kbd_mode -u
```

NAME

pcitweak - read/write PCI config space

SYNOPSIS

pcitweak −l
pcitweak −r *PCI-ID* [**−b**|**−h**] *offset*
pcitweak −w *PCI-ID* [**−b**|**−h**] *offset value*

DESCRIPTION

Pcitweak is a utility that can be used to examine or change registers in the PCI configuration space. On most platforms *pcitweak* can only be run by the root user.

OPTIONS

−l Probe the PCI buses and print a line for each detected device. Each line contains the bus location (bus:device:function), chip vendor/device, card (subsystem) vendor/card, revision, class and header type. All values printed are in hexadecimal.

−r *PCI-ID*
Read the PCI configuration space register at *offset* for the PCI device at bus location *PCI-ID*. *PCI-ID* should be given in the form bus:device:function, with each value in hexadecimal. By default, a 32-bit register is read.

−w *PCI-ID*
Write *value* to the PCI configuration space register at *offset* for the PCI device at bus location *PCI-ID*. *PCI-ID* should be given in the form bus:device:function, with each value in hexadecimal. By default, a 32-bit register is written.

−b Read or write an 8-bit value (byte).

−h Read or write a 16-bit value (halfword).

SEE ALSO

scanpci(1)

AUTHORS

David Dawes (*dawes@xfree86.org*).

NAME

scanpci - scan/probe PCI buses

SYNOPSIS

scanpci [**−v12OfV**]

DESCRIPTION

Scanpci is a utility that can be used to scan PCI buses and report information about the configuration space settings for each PCI device. On most platforms, *scanpci* can only be run by the root user.

OPTIONS

−v Print the configuration space information for each device in a verbose format. Without this option, only a brief description is printed for each device.

−1 Use PCI config type 1.

−2 Use PCI config type 2.

−f Used in conjunction with the above two options, this forces the specified configuration type to be used for config space access.

−O Use the OS's PCI config space access mechanism to access the PCI config space (when available).

−V *n* Set the verbosity level to *n* for the internal PCI scanner. This is primarily for debugging use.

SEE ALSO

pcitweak(1)

AUTHORS

NAME

fbdevhw − os-specific submodule for framebuffer device access

DESCRIPTION

fbdevhw provides functions for talking to a framebuffer device. It is os-specific. It is a submodule used by other video drivers. A **fbdevhw** module is currently available for linux framebuffer devices.

fbdev(4) is a non-accelerated driver which runs on top of the fbdevhw module. fbdevhw can be used by other drivers too, this is usually activated with ‘Option "UseFBDev"’ in the device section.

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7), fbdev(4)

AUTHORS

Authors include: Gerd Knorr, based on the XF68_FBDev Server code (Martin Schaller, Geert Uytterhoeven).

NAME

acecad − Acecad Flair input driver

SYNOPSIS

Section "InputDevice"
Identifier "*idevname***"**
Driver "acecad"
Option "Device" "*devpath***"**
...
EndSection

DESCRIPTION

acecad is an XFree86 input driver for Acecad Flair devices...

The **acecad** driver functions as a pointer input device, and may be used as the X server's core pointer. THIS MAN PAGE NEEDS TO BE FILLED IN.

SUPPORTED HARDWARE

What is supported...

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details and for options that can be used with all input drivers. This section only covers configuration details specific to this driver.

Config details...

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7).

AUTHORS

Authors include... Edouard TISSERANT

NAME

calcomp − Calcomp input driver

SYNOPSIS

Section "InputDevice"
Identifier "*idevname***"**
Driver "calcomp"
Option "Device" "*devpath***"**
...
EndSection

DESCRIPTION

calcomp is an XFree86 input driver for Calcomp devices.

The **calcomp** driver functions as a pointer input device, and may be used as the X server's core pointer.

SUPPORTED HARDWARE

This driver supports the Calcomp binary format used by the Drawing Board II and III series.

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details and for options that can be used with all input drivers. This section only covers configuration details specific to this driver.

Both the 3 button stylus and the 4- or 16 button lens cursors can be used without changing the configuration file. Support for pressure sensitivity has not been tested, so the solid-tip stylus will probably not work.

This device supports the following entries:

Option *"Device" "path"*
sets the path to the special file which represents the serial line where the tablet is plugged. This option is mandatory.

Option *"Cursor" "Stylus"*|*"Puck"*
this option is supported for backward compatibility only, but it should not be necessary.

Option *"DeviceName" "name"*
sets the name of the X device. Some user-space programs may require a fixed name, e.g. TABLET, to recognize the digitizer.

Option *"Mode" "Relative"*|*"Absolute"*
sets the mode of the device. Currently only Absolute mode is supported.

Option *"Pressure" "on"*
enables pressure reporting if your tablet supports it. This option is untested and may not work.

Option *"AlwaysCore" "on"*
enables the sharing of the core pointer. When this feature is enabled, the device will take control of the core pointer (and thus will emit core events) and at the same time will be able, when asked so, to report extended events.

Option *"MinX" "number"*
X coordinate of the bottom left corner of the active zone.

Option *"MinY" "number"*
Y coordinate of the bottom left corner of the active zone.

Option *"MaxX" "Inumber"*
X coordinate of the top right corner of the active zone.

Option *"MaxY" "number"*
Y coordinate of the top right corner of the active zone.

Option *"DebugLevel" number*
sets the level of debugging info reported.

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7).

AUTHORS

Martin Kroeker <mk@daveg.com>

NAME

citron − Citron Infrared Touch Driver (CiTouch)

SYNOPSIS

Section "InputDevice"
Identifier "*idevname***"**
Driver "citron"
Option "Device" "*devpath***"**
...
EndSection

DESCRIPTION

citron is a XFree86 input driver for *Citron Infrared Touch* devices.

The **citron** driver acts as a pointer input device, and may be used as the X server's core pointer. It is connected via a "RS232" with the host.

SUPPORTED HARDWARE

At the moment the following touches are supported. They are also available as *ZPress* touches.

IRT6I5-V2.x
6.5 inch Infrared Touch

IRT10I4-V4.x
10.4 inch Infrared Touch

IRT12I1-V2.x
12.1 inch Infrared Touch

IRT15I1-V1.x
15.1 inch Infrared Touch

CONFIGURATION DETAILS

Please refer to XF86Config(5x) for general configuration details and for options that can be used with all input drivers. This section only covers configuration details specific to this driver. For better understanding please read also the **CTS** and various **IRT** manuals which are available in "pdf" format from Citron web page **www.citron.de** or directly from Citron.

The following driver **Options** are supported:

Option "Device" "*devpath***"**
Specify the device path for the citron touch. Valid devices are:

/dev/ttyS0, /dev/ttyS1, ... This option is mandatory.

It's important to specify the right device Note: com1 -> /dev/ttyS0, com2 -> /dev/ttyS1 ....

Option "ScreenNumber" "*screennumber***"**
sets the *screennumber* for the *citron* InputDevice.

Default: ScreenNumber: "0"

Option "MinX, MinY" "*value***"**
These are the minimum X and Y values for the *citron* input device.

Note: MinX, MinY must be less than MaxX, MaxY.

Range: "0" - "65535"

Default: MinX: "0" MinY: "0"

Option "MaxX, MaxY" "*value*"

These are the maximum X and Y values for the *citron* input device.

Note: MaxX, MaxY must be greater than MinX, MinY.

Range: "0" - "65535"

Default: MaxX: "65535" MaxY: "65535"

Option "ButtonNumber" "*value*"

This value is responsible for the *button number* that is returned within the xf86PostButton event message

Range: "0" - "255"

Default: "1"

Option "ButtonThreshold" "*value*"

This value is responsible for the *button threshold.* It changes the pressure sensitivity of the touch. A higher number corresponds to a higher pressure.

Note: This feature is only available with pressure sensitive hardware.

Range: "0" - "255"

Default: "20"

Sleep-Mode

If the IRT is in *Doze-Mode* and Touch Zone is not interrupted for another certain span of time, the so-called *Sleep-Mode* is activated. The *Sleep-Mode* decreases the scan rate of the beams even further than the *Doze-Mode* does (see below). This way the life expectancy of the beams is prolonged and the power consumption of the IRT is reduced. As soon as an interruption of the Touch Zone is detected, the *Sleep-Mode* is deactivated and the Touch Zone will again be scanned with the maximum speed. With the Sleep-Mode activated, depending on the set scan rate the IRT's response time can be considerably longer as in normal operation. If, for example, a scan rate of 500 ms / scan is set, it may last up to a half of a second until the IRT detects the interruption and deactivates the *Sleep-Mode.*

Option "SleepMode" "*mode*"

This value is responsible for the *sleep-mode* of the touch.
Determines the behaviour of the Sleep-Mode.

0x00
No message at either activation or deactivation

0x01
Message at activation

0x02
Message at deactivation

0x03
Message at activation and deactivation

0x10 GP_OUT output set according to the Sleep-Mode status

Values: "0" "1" "2" "3" "16"

Default: "0"

Option "SleepTime" "*time***"**

This value is responsible for the *sleep-time* of the touch. It is the activation time in seconds ("0" = immediately activated, "65535" = always deactivated).

Range: "0" - "65535" [s]

Default: "65535" => deactivated

Option "SleepScan" "*scan***"**

This value is responsible for the *scan-time* of the touch. This is the time interval between two scan operations while in Sleep-Mode. The time interval is set in steps of milliseconds.

Range: "0" - "65535" [ms]

Default: "500"

Option "PWMAdjSrc" "*value***"**

Option "PWMAdjDst" "*value***"**

These parameters are used to adjust the brightness of different backlight inverters. At the moment 2 backlight inverters are used: 0=TDK 1=AC. If you want a AC backlight inverter to behave like an AC type you have to set *PWMAdjSrc* to 0 (TDK) and *PWMAdjDst* to 1 (AC).

Range: "0" - "1"

Default: "-1" (no adjustment)

Option "PWMActive" "*value***"**

This value determines the mark-to-space ratio of the *PWM* output while in normal operation (sleep-mode not active). Higher values result in longer pulse widths. This output signal can be used in conjunction with the *Citron AWBI* to do backlight-dimming via the touch.

Range: "0" - "255"

Default: "255" (max. brightness)

Option "PWMSleep" "*value***"**

This value determines the mark-to-space ratio of the *PWM* output while in sleep-mode (-> *Sleep-Mode, SleepScan, SleepTime*) operation (sleep-mode active). Higher values result in longer pulse widths.

Range: "0" - "255"

Default: "255" (max. brightness)

Option "PWMFreq" "*value***"**

This value determines the *PWM* frequency in Hertz

Range: "39" - "9803"

Default: "9803" (max. frequency)

Option "ClickMode" "*mode*"

With mode one can select between 5 *ClickModes*

"1" = ClickMode Enter

With this mode every interruption of the infrared beams will activate a ButtonPress event and after the interruption a ButtonRelease event will be sent.

"2" = ClickMode Dual

With this mode every interruption will sent a Proximity event and every second interruption a ButtonPress event. With the release of the interruption (while one interruption is still active) a ButtonRelease event will be sent.

"3" = ClickMode Dual Exit

With this mode every interruption will sent a ProximityIn event and every second interruption a ButtonPress event. With the release of the interruption (while one interruption is still active) no ButtonRelease event will be sent. Only if all interruptions are released a ButtonRelease followed by a ProximityOut event will be sent.

"4" = ClickMode ZPress

With this mode every interruption will sent a ProximityIn event. Only if a certain pressure is exceeded a ButtonPress event will occur. If the pressure falls below a certain limit a ButtonRelease event will be sent. After also the interruption is released a ProximityOut event is generated.

"5" = ClickMode ZPress Exit

This mode is similat to "Clickmode Dual Exit". The first interruption of the beams will sent a ProximityIn event. Only if a certain pressure is exceeded a ButtonPress event will occur. If the pressure falls below a certain limit no ButtonRelease event will be sent. After the interruption is also released a ButtonRelease followed by a ProximityOut event is generated.

Range: "1" - "5"

Default: "1" (ClickMode Enter)

Option "Origin" "*value*"

This value sets the coordinates origin to one of the four corners of the screen. The following values are accepted: "0" TOPLEFT: Origin set to the left-hand side top corner. "1" TOPRIGHT: Origin set to the right-hand side top corner. "2" BOTTOMRIGHT: Origin set to the right-hand side bottom corner. "3" BOTTOMLEFT: Origin set to the left-hand side bottom corner.

Range: "0" - "3"

Default: "0" (TOPLEFT)

Doze-Mode

If for a certain span of time the Touch Zone is not interrupted, the so-called Doze-Mode is automatically activated. The activated Doze-Mode slightly decreases the scan rate of the beams. This way the power consumption of the IRT is reduced. As soon as an interruption of the Touch Zone is detected, the Doze-Mode is deactivated and the Touch Zone will again be scanned with the maximum speed.

Option "DozeMode" "*mode*"

This value is responsible for the *doze-mode* of the touch.

Determines the behaviour of the Doze-Mode.

0x00 No message at either activation or deactivation

0x01 Message at activation

0x02 Message at deactivation

0x03 Message at activation and deactivation

0x10 GP_OUT output set according to the Doze-Mode status

If the GP_OUT output is already controlled by the *Sleep-Mode* it is no longer available as an output port anymore.

Values: "0" "1" "2" "3" "16"

Default: "0"

Option "DozeTime" "*time*"

This value is responsible for the *doze-time* of the touch. It is the activation time in seconds ("0" = immediately activated, "65535" = always deactivated).

Range: "0" - "65535" [s]

Default: "65535" => deactivated

Option "DozeScan" "*scan*"

This value is responsible for the *scan-time* of the touch. This is the time interval between two scan operations while in Doze-Mode. The time interval is set in steps of milliseconds.

Range: "0" - "65535" [ms]

Default: "500"

Option "DeltaX" "*value*"

This value determines a virtual area at the left and right side of the current cursor position where the cursor didn't move. Within this area no "MotionNotify" event will be sent.

Range: "0" - "255"

Default: "0" (no deltaX)

Option "DeltaY" "*value*"

This value determines a virtual area at the top and bottom of the current cursor position where the cursor didn't move. Within this area no "MotionNotify" event will be sent.

Range: "0" - "255"

Default: "0" (no deltaY)

Option "Beep" "*value*"
This value determines if a "ButtonPress" and/or a "ButtonRelease" event should sound the buzzer. "0" deactivates the buzzer while every other value will activate it.

Range: "0" - "1"

Default: "0" (deactivated)

Option "PressVol" "*value*"
This value determines the volume of the buzzer (0-100%) when a "ButtonPress" event is sent.

Range: "0" - "100"

Default: "100"

Option "PressPitch" "*value*"
This value determines the pitch of the tone when a "ButtonPress" event is sent.

Range: "0" - "3000"

Default: "880"

Option "PressDur" "*value*"
This value determines the duration of the tone in ms when a "ButtonPress" event is sent.

Range: "0" - "255"

Default: "15"

Option "ReleaseVol" "*value*"
This value determines the volume of the buzzer (0-100%) when a "ButtonRelease" event is sent.

Range: "0" - "100"

Default: "100"

Option "ReleasePitch" "*value*"
This value determines the pitch of the tone when when a "ButtonRelease" event is sent.

Range: "0" - "3000"

Default: "1200"

Option "ReleseDur" "*value*"
This value determines the duration of the tone in ms when when a "ButtonRelease" event is sent.

Range: "0" - "255"

Default: "10"

Option "BeamTimeout" "*value*"
Determines the time span in seconds, that has to elapse before a beam is considered defective, blanked-out and excluded from the coordinates evaluation.

Range: "0" - "65535"

Default: "30" (30 seconds)

Option "TouchTime" "*value*"
Determines the minimum time span in steps of 10ms for a valid interruption. In order for an interruption to be reported to the host computer as valid, it needs to remain at the same spot for at least the time span declared here.

Range: "0" - "255"

Default: "0" (=6,5 ms)

Option "EnterCount" "*count*"
Number of skipped "enter reports". Reports are sent approx. every 20ms.

Range: "0" - "31"

Default: "3" (3 skipped messages = 60ms)

Option "ZEnterCount" "*count*"
Number of skipped "enter reports" while in pressure sensitive mode. Reports are sent approx. every 20ms.

Range: "0" - "31"

Default: "1" (1 skipped messages = 20ms)

Option "LockZEnterTime" "*count*"
Minimum duration of an AreaPressEnter state (Pressure > AreaPressure) before a PressEnter event is issued. The time is given in 10ms steps.

Range: "0" - "255"

Default: "1" (10ms)

Option "LockZExitTime" "*count*"
Minimum duration of an AreaPressExit state (Pressure < AreaPressure/2) before a PressExit event is issued. The time is given in 10ms steps.

Range: "0" - "255"

Default: "1" (10ms)

Option "LockZLockTime" "*count*"
Minimum gap between a PressExit and a PressEnter event. The time is in 10ms steps.

Range: "0" - "255"

Default: "10" (100ms)

Option "DualCount" "*count*"
Number of skipped "dual touch error". Reports are sent approx. every 20ms. This option is only available for "ZPress" and "ZPress Exit" modes.

Range: "0" - "31"

Default: "2" (2 skipped messages = 40ms)

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7).

AUTHORS

2000-2003 - written by Citron GmbH (support@citron.de)

NAME

dmc − DMC input driver

SYNOPSIS

Section "InputDevice"
Identifier "*idevname***"**
Driver "dmc"
Option "Device" "*devpath***"**
...
EndSection

DESCRIPTION

dmc is an XFree86 input driver for DMC FIT10-controller...

The **dmc** driver functions as a pointer input device, and may be used as the X server's core pointer. THIS MAN PAGE NEEDS TO BE FILLED IN.

SUPPORTED HARDWARE

What is supported...

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details and for options that can be used with all input drivers. This section only covers configuration details specific to this driver.

Config details...

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7).

AUTHORS

Authors include...

NAME

dynapro − Dynapro input driver

SYNOPSIS

Section "InputDevice"
  Identifier "*idevname***"**
  Driver "dynapro"
  Option "Device" "*devpath***"**
  ...
EndSection

DESCRIPTION

dynapro is an XFree86 input driver for Dynapro devices...

The **dynapro** driver functions as a pointer input device, and may be used as the X server's core pointer. THIS MAN PAGE NEEDS TO BE FILLED IN.

SUPPORTED HARDWARE

What is supported...

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details and for options that can be used with all input drivers. This section only covers configuration details specific to this driver.

Config details...

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7).

AUTHORS

Authors include...

NAME

elographics − Elographics input driver

SYNOPSIS

Section "InputDevice"
Identifier "*idevname***"**
Driver "elographics"
Option "Device" "*devpath***"**
...
EndSection

DESCRIPTION

elographics is an XFree86 input driver for Elographics devices...

The **elographics** driver functions as a pointer input device, and may be used as the X server's core pointer. THIS MAN PAGE NEEDS TO BE FILLED IN.

SUPPORTED HARDWARE

What is supported...

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details and for options that can be used with all input drivers. This section only covers configuration details specific to this driver.

Config details...

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7).

AUTHORS

Authors include...
Patrick Lecoanet

NAME

fpit − Fujitsu Stylistic input driver

SYNOPSIS

Section "InputDevice"
  Identifier "*idevname***"**
  Driver "fpit"
  Option "Device" "*devpath***"**
  ...
EndSection

DESCRIPTION

fpit is an XFree86 input driver for Fujitsu Stylistic Tablet PCs.

The **fpit** driver functions as a pointer input device, and may be used as the X server's core pointer.

SUPPORTED HARDWARE

This driver supports the touchscreen of the Stylistic LT and (with special options) of the Stylistic 500, 1000 and 2300.

Under Linux the Fujitsus serial port is not, by default, detected. Therefore the following must be added to one of your start-up scripts. (Either one of the X scripts, or to rc.local or similar).

setserial /dev/ttyS3 autoconfig

setserial /dev/ttyS3 IRQ 15 baud_base 115200 port 0xfce8

CONFIGURATION DETAILS

Please refer to XF86Config(5x) for general configuration details and for options that can be used with all input drivers. This section only covers configuration details specific to this driver.

The device supports the following options:

Option *"MaximumXPosition" "number"*
Sets the maximum X position, use this to callibrate your touchscreen's right hand edge.

Option *"MinimumXPosition" "number"*
Sets the minimum X position, use this to callibrate your touchscreen's left hand edge.

Option *"MaximumYPosition" "number"*

Option *"MinimumYPosition" "number"*
Same as for X axis, but for Y axis.

Option *"InvertX"*

Option *"InvertY"*
Invert the specified axis.

Option *"SwapXY"*
Swap the X and Y axis.

Option *"Rotate" "CW"*
Option *"Rotate" "CWW"* Manipulate the invert and swap options to match screen rotations.

Option *"DeviceName" "name"*
Option *"DeviceName" "name"* sets the name of the X device.

Option *"AlwaysCore" "on"*
enables the sharing of the core pointer. When this feature is enabled, the device will take control of the core pointer (and thus will emit core events) and at the same time will be able, when asked so, to report extended events. You can use the last available integer feedback to control this feature. When the value of the feedback is zero, the feature is disabled. The

feature is enabled for any other value.

Option *"DebugLevel" number*
sets the level of debugging info reported.

Option *"BaudRate" "38400"***,** *"19200"* **or** *"9600"* **(default)**
changes the serial link speed.

Example, for Stylistic LT setup is:
Section "InputDevice"
Identifier "*mouse0***"**
Driver "fpit"
Option "Device" "*/dev/ttyS3***"**
EndSection

And for other Stylistic devices try:
Section "InputDevice"
Identifier "*mouse0***"**
Driver "fpit"
Option "Device" "*/dev/ttyS3***"**
Option "BaudRate" "*19200***"**
Option "MaximumXPosition" "*6250***"**
Option "MaximumYPosition" "*4950***"**
Option "MinimumXPosition" "*130***"**
Option "MinimumYPosition" "*0***"**
Option "InvertY"
EndSection

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7).

AUTHORS

Original FPIT port: Rob Tsuk <rob@tsuk.com> and John Apfelbaum <johnapf@linuxslate.com>

X4 Port: Richard Miller-Smith <richard.miller-smith@philips.com>, based on Elographics code from: Patrick Lecoanet

X4.2 Cleanup: Alan Cox

NAME

js_x − JamStudio input driver

SYNOPSIS

Section "InputDevice"
Identifier "*devname***"**
Driver "js_x"
Option "Device" "*devpath***"**
Option "MaxX" "*int***"**
Option "MaxY" "*int***"**
Option "MinX" "*int***"**
Option "MinY" "*int***"**
Option "PressMax" "*int***"**
Option "PressMin" "*int***"**
Option "PressDiv" "*int***"**
EndSection

DESCRIPTION

js_x is an XFree86 input driver for JamStudio devices.

The **js_x** driver functions as a pointer input device, and may be used as the X server's core pointer.

SUPPORTED HARDWARE

This driver supports the KB-Gear JamStudio pentablet. This X-Input driver should work on any OS supporting the hiddev raw USB HID driver.

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details and for options that can be used with all input drivers. This section only covers configuration details specific to this driver.

Option *"Device" "path"*
sets the path to the raw HID device to which the tablet was assigned. This option is mandatory.

Option *"MinX" "int"*

Option *"MaxX" "int"*

Option *"MinY" "int"*

Option *"MaxY" "int"*
sets the minimum and maximum values returned for the absolute X,Y axis of the pen tablet. These values default to 0-8000 for X and 0-6000 for Y. It should generally be safe to leave these values untouched.

Option *"PressMin "int"*

Option *"PressMax "int"*
sets the minimum and maximum values returned for the pressure sensitive tip. These values default to 0-127. It should generally be safe to leave these values untouched.

Option *"PressDiv" "int"*
sets the divider for the returned pressure value. This option will allow you to return a smaller set of values for the pressure sensitive tip allowing for finer control. The returned value is computed as follows:

X / PressDiv = returned value
where X equals the value read from the tablet.

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7).

AUTHORS

Brian Goines <bgoines78@comcast.net>

NAME

kbd − Keyboard input driver

SYNOPSIS

Section "InputDevice"
Identifier "*idevname***"**
Driver "kbd"
...
EndSection

DESCRIPTION

kbd is an XFree86 input driver for keyboards. The driver supports the standard OS-provided keyboard interface, but these are currently only available to this driver module for Linux and BSD. This driver is experimental, but will soon replace the built-in **keyboard** driver.

The **kbd** driver functions as a keyboard input device, and may be used as the X server's core keyboard.

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details and for options that can be used with all input drivers. This section only covers configuration details specific to this driver.

The following driver **Options** are supported:

Option "Device" "*string***"**
Specify the keyboard device. Default: the OS's default console keyboard input source.

Option "Protocol" "*string***"**
Specify the keyboard protocol. Valid protocol types include:

Standard, Xqueue.

Not all protocols are supported on all platforms. Default: "Standard".

Option "AutoRepeat" "*delay rate***"**
sets the auto repeat behaviour for the keyboard. This is not implemented on all platforms. *delay* is the time in milliseconds before a key starts repeating. *rate* is the number of times a key repeats per second. Default: "500 30".

Option "XLeds" "*ledlist***"**
makes the keyboard LEDs specified in *ledlist* available for client use instead of their traditional function (Scroll Lock, Caps Lock and Num Lock). The numbers in the list are in the range 1 to 3. Default: empty list.

Option "XkbRules" "*rules***"**
specifies which XKB rules file to use for interpreting the **XkbModel**, **XkbLayout**, **XkbVariant**, and **XkbOptions** settings. Default: "xfree86" for most platforms, but "xfree98" for the Japanese PC-98 platforms.

Option "XkbModel" "*modelname***"**
specifies the XKB keyboard model name. Default: "pc101" for most platforms, but "pc98" for the Japanese PC-98 platforms, and "pc101_sol8x86" for Solaris 8 on x86.

Option "XkbLayout" "*layoutname***"**
specifies the XKB keyboard layout name. This is usually the country or language type of the keyboard. Default: "us" for most platforms, but "nec/jp" for the Japanese PC-98 platforms.

Option "XkbVariant" "*variants***"**
specifies the XKB keyboard variant components. These can be used to enhance the keyboard layout details. Default: not set.

Option "XkbOptions" "*options***"**
specifies the XKB keyboard option components. These can be used to enhance the keyboard behaviour. Default: not set.

Some other XKB-related options are available, but they are incompatible with the ones listed above and are not recommended, so they are not documented here.

SEE ALSO

keyboard(4), XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7).

NAME

keyboard − Keyboard input driver

SYNOPSIS

Section "InputDevice"
Identifier "*idevname***"**
Driver "keyboard"
...
EndSection

DESCRIPTION

keyboard is an XFree86 input driver for keyboards. The driver supports the standard OS-provided keyboard interface. This driver is currently built-in to the core X server.

The **keyboard** driver functions as a keyboard input device, and may be used as the X server's core keyboard. This driver is currently built-in to the core X server, and multiple instances are not yet supported.

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details and for options that can be used with all input drivers. This section only covers configuration details specific to this driver.

The following driver **Options** are supported:

Option "Protocol" "*string***"**
Specify the keyboard protocol. Valid protocol types include:

Standard, Xqueue.

Not all protocols are supported on all platforms. Default: "Standard".

Option "AutoRepeat" "*delay rate***"**
sets the auto repeat behaviour for the keyboard. This is not implemented on all platforms. *delay* is the time in milliseconds before a key starts repeating. *rate* is the number of times a key repeats per second. Default: "500 30".

Option "XLeds" "*ledlist***"**
makes the keyboard LEDs specified in *ledlist* available for client use instead of their traditional function (Scroll Lock, Caps Lock and Num Lock). The numbers in the list are in the range 1 to 3. Default: empty list.

Option "XkbDisable" "*boolean***"**
disable/enable the XKEYBOARD extension. The −kb command line option overrides this config file option. Default: XKB is enabled.

NOTE: This option should be specified in the **ServerFlags** section rather than here. It's use here is deprecated.

Option "XkbRules" "*rules***"**
specifies which XKB rules file to use for interpreting the **XkbModel**, **XkbLayout**, **XkbVariant**, and **XkbOptions** settings. Default: "xfree86" for most platforms, but "xfree98" for the Japanese PC-98 platforms.

Option "XkbModel" "*modelname***"**
specifies the XKB keyboard model name. Default: "pc101" for most platforms, but "pc98" for the Japanese PC-98 platforms, and "pc101_sol8x86" for Solaris 8 on x86.

Option "XkbLayout" "*layoutname***"**
specifies the XKB keyboard layout name. This is usually the country or language type of the keyboard. Default: "us" for most platforms, but "nec/jp" for the Japanese PC-98 platforms.

Option "XkbVariant" "*variants***"**
specifies the XKB keyboard variant components. These can be used to enhance the keyboard layout details. Default: not set.

Option "XkbOptions" "*options*"
specifies the XKB keyboard option components. These can be used to enhance the keyboard behaviour. Default: not set.

Some other XKB-related options are available, but they are incompatible with the ones listed above and are not recommended, so they are not documented here.

SEE ALSO

kbd(4), XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7).

NAME

void − null input driver

SYNOPSIS

Section "InputDevice"
Identifier "*idevname***"**
Driver "magictouch"
...
EndSection

DESCRIPTION

magictouch is an XFree86 input driver.

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details and for options that can be used with all input drivers. This driver doesn't have any configuration options in addition to those.

SEE ALSO

XFree86(1), XF86Config(5), xf86cfg(1), xf86config(1), Xserver(1), X(7).

AUTHORS

Authors include...

NAME

microtouch − MicroTouch input driver

SYNOPSIS

Section "InputDevice"
  Identifier "*idevname***"**
  Driver "microtouch"
  Option "Device"   "*devpath***"**
  ...
EndSection

DESCRIPTION

microtouch is an XFree86 input driver for MicroTouch devices...

The **microtouch** driver functions as a pointer input device, and may be used as the X server's core pointer. THIS MAN PAGE NEEDS TO BE FILLED IN.

SUPPORTED HARDWARE

What is supported...

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details and for options that can be used with all input drivers. This section only covers configuration details specific to this driver.

Config details...

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7).

AUTHORS

Authors include...

NAME

mouse − Mouse input driver

SYNOPSIS

Section "InputDevice"
Identifier "*idevname***"**
Driver "mouse"
Option "Protocol" "*protoname***"**
Option "Device" "*devpath***"**
...
EndSection

DESCRIPTION

mouse is an XFree86 input driver for mice. The driver supports most available mouse types and interfaces. USB mice are only supported on some OSs, and the level of support for PS/2 mice depends on the OS.

The **mouse** driver functions as a pointer input device, and may be used as the X server's core pointer. Multiple mice are supported by multiple instances of this driver.

SUPPORTED HARDWARE

There is a detailed list of hardware that the **mouse** driver supports in the *README.mouse* document. This can be found in /usr/X11R6/lib/X11/doc/, or online at http://www.xfree86.org/current/mouse.html.

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details and for options that can be used with all input drivers. This section only covers configuration details specific to this driver.

The driver can auto-detect the mouse type on some platforms On some platforms this is limited to plug and play serial mice, and on some the auto-detection works for any mouse that the OS's kernel driver supports. On others, it is always necessary to specify the mouse protocol in the config file. The *README.mouse* document contains some detailed information about this.

The following driver **Options** are supported:

Option "Protocol" "*string***"**
Specify the mouse protocol. Valid protocol types include:

Auto, Microsoft, MouseSystems, MMSeries, Logitech, MouseMan, MMHitTab, GlidePoint, IntelliMouse, ThinkingMouse, ValuMouseScroll, AceCad, PS/2, ImPS/2, ExplorerPS/2, ThinkingMousePS/2, MouseManPlusPS/2, GlidePointPS/2, NetMousePS/2, NetScrollPS/2, BusMouse, SysMouse, WSMouse, USB, Xqueue.

Not all protocols are supported on all platforms. The "Auto" platform specifies that protocol auto-detection should be attempted. There is no default protocol setting, and specifying this option is mandatory.

Option "Device" "*string***"**
Specifies the device through which the mouse can be accessed. A common setting is "/dev/mouse", which is often a symbolic link to the real device. This option is mandatory, and there is no default setting.

Option "Buttons" "*integer***"**
Specifies the number of mouse buttons. In cases where the number of buttons cannot be auto-detected, the default value is 3.

Option "Emulate3Buttons" "*boolean***"**
Enable/disable the emulation of the third (middle) mouse button for mice which only have two physical buttons. The third button is emulated by pressing both buttons simultaneously. Default: off

Option "Emulate3Timeout" "*integer***"**
Sets the timeout (in milliseconds) that the driver waits before deciding if two buttons where pressed "simultaneously" when 3 button emulation is enabled. Default: 50.

Option "ChordMiddle" "*boolean*"
Enable/disable handling of mice that send left+right events when the middle button is used. Default: off.

Option "EmulateWheel" "*boolean*"
Enable/disable "wheel" emulation. Wheel emulation means emulating button press/release events when the mouse is moved while a specific real button is pressed. Wheel button events (typically buttons 4 and 5) are usually used for scrolling. Wheel emulation is useful for getting wheel-like behaviour with trackballs. It can also be useful for mice with 4 or more buttons but no wheel. See the description of the **EmulateWheelButton**, **EmulateWheelInertia**, **XAxisMapping**, and **YAxisMapping** options below. Default: off.

Option "EmulateWheelButton" "*integer*"
Specifies which button must be held down to enable wheel emulation mode. While this button is down, X and/or Y pointer movement will generate button press/release events as specified for the **XAxisMapping** and **YAxisMapping** settings. Default: 4.

Option "EmulateWheelInertia" "*integer*"
Specifies how far (in pixels) the pointer must move to generate button press/release events in wheel emulation mode. Default: 50.

Option "XAxisMapping" "*N1 N2*"
Specifies which buttons are mapped to motion in the X direction in wheel emulation mode. Button number *N1* is mapped to the negative X axis motion and button number *N2* is mapped to the positive X axis motion. Default: no mapping.

Option "YAxisMapping" "*N1 N2*"
Specifies which buttons are mapped to motion in the Y direction in wheel emulation mode. Button number *N1* is mapped to the negative Y axis motion and button number *N2* is mapped to the positive Y axis motion. Default: "4 5".

Option "ZAxisMapping" "X"

Option "ZAxisMapping" "Y"

Option "ZAxisMapping" "*N1 N2*"

Option "ZAxisMapping" "*N1 N2 N3 N4*"
Set the mapping for the Z axis (wheel) motion to buttons or another axis (**X** or **Y**). Button number *N1* is mapped to the negative Z axis motion and button number *N2* is mapped to the positive Z axis motion. For mice with two wheels, four button numbers can be specified, with the negative and positive motion of the second wheel mapped respectively to buttons number *N3* and *N4*. Default: no mapping.

Option "FlipXY" "*boolean*"
Enable/disable swapping the X and Y axes. This transformation is applied after the **InvX**, **InvY** and **AngleOffset** transformations. Default: off.

Option "InvX" "*boolean*"
Invert the X axis. Default: off.

Option "InvY" "*boolean*"
Invert the Y axis. Default: off.

Option "AngleOffset" "*integer*"
Specify a clockwise angular offset (in degrees) to apply to the pointer motion. This transformation is applied before the **FlipXY**, **InvX** and **InvY** transformations. Default: 0.

Option "SampleRate" "*integer*"
Sets the number of motion/button events the mouse sends per second. Setting this is only supported for some mice, including some Logitech mice and some PS/2 mice on some platforms. Default: whatever the mouse is already set to.

Option "Resolution" "*integer***"**
Sets the resolution of the device in counts per inch. Setting this is only supported for some mice, including some PS/2 mice on some platforms. Default: whatever the mouse is already set to.

Option "DragLockButtons" "*L1 B2 L3 B4***"**
Sets "drag lock buttons" that simulate holding a button down, so that low dexterity people do not have to hold a button down at the same time they move a mouse cursor. Button numbers occur in pairs, with the lock button number occurring first, followed by the button number that is the target of the lock button.

Option "DragLockButtons" "*M1***"**
Sets a "master drag lock button" that acts as a "Meta Key" indicating that the next button pressed is to be "drag locked".

Option "ClearDTR" "*boolean***"**
Enable/disable clearing the DTR line on the serial port used by the mouse. Some dual-protocol mice require the DTR line to be cleared to operate in the non-default protocol. This option is for serial mice only. Default: off.

Option "ClearRTS" "*boolean***"**
Enable/disable clearing the RTS line on the serial port used by the mouse. Some dual-protocol mice require the RTS line to be cleared to operate in the non-default protocol. This option is for serial mice only. Default: off.

Option "BaudRate" "*integer***"**
Set the baud rate to use for communicating with a serial mouse. This option should rarely be required because the default is correct for almost all situations. Valid values include: 300, 1200, 2400, 4800, 9600, 19200. Default: 1200.

There are some other options that may be used to control various parameters for serial port communication, but they are not documented here because the driver sets them correctly for each mouse protocol type.

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7), README.mouse.

NAME

mutouch − Microtouch input driver

SYNOPSIS

Section "InputDevice"
Identifier "*idevname***"**
Driver "mutouch"
Option "Device" "*devpath***"**
...
EndSection

DESCRIPTION

mutouch is an XFree86 input driver for Microtouch devices...

The **mutouch** driver functions as a pointer input device, and may be used as the X server's core pointer. THIS MAN PAGE NEEDS TO BE FILLED IN.

SUPPORTED HARDWARE

What is supported...

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details and for options that can be used with all input drivers. This section only covers configuration details specific to this driver.

Config details...

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7).

AUTHORS

Authors include...
Patrick Lecoanet

NAME

palmax − Palmax (TR88L803) touchscreen driver

SYNOPSIS

Section "InputDevice"
Identifier "*idevname***"**
Driver "palmax"
Option "Device" "*devpath***"**
...
EndSection

DESCRIPTION

palmax is an XFree86 input driver for the Palmax PD1000/PD1100

The **palmax** driver functions as a pointer input device, and is normally used as the X server's core pointer. It supports positioning and mouse buttons using the touchscreen display and lid buttons on the Palmax machines.

SUPPORTED HARDWARE

Palmax PD1000, Palmax PD1100. In theory also any other system using a TR88L803 wired to a serial port.

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details and for options that can be used with all input drivers. This section only covers configuration details specific to this driver.

The following driver **options** are supported

Option "MinX" "*integer***"**
Set the left hand X value from the touchscreen, for calibration.

Option "MaxX" "*integer***"**
Set the right hand X value from the touchscreen, for calibration.

Option "MinY" "*integer***"**
Set the top Y value from the touchscreen, for calibration.

Option "MaxY" "*integer***"**
Set the bottom Y value from the touchscreen, for calibration.

Option "Screen" "*integer***"**
The screen to attach to the touchscreen when running with multiple screens. The default is screen 0.

Option "Device" "*string***"**
The serial port that is attached to the touchscreen interface. On the Palmax PD1000 and PD1100 this is ttyS0.

Option "DeviceName" "*string***"**
Set the X11 device name for the touchscreen. This defaults to TOUCHSCREEN.

Option "PortraitMode" "*string***"**
Set the display orientation. The default is "landscape" but you can rotate the screen clockwise ("portrait") or anticlockwise ("portraitCCW").

Option "SwapXY" "*boolean***"**
Swap the X and Y values on the display. The default is false.

Option "TapButton" "*boolean***"**
Set the touchscreen tap to act as mouse button 1. This allows single handed operation except when using the menu buttons. The default is false.

BUGS

The driver has been tested on the Palmax systems, the defaults reflect the Palmax hardware and should work out of the box. No testing has been done on other systems using the same digitizer.

Support for a double-tap menu button option would be nice.

The smoothing algorithm would benefit from real mathematics.

XFree86 needs a nice calibration tool.

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7).

AUTHORS

Authors include...
Alan Cox

NAME

penmount – PenMount input driver

SYNOPSIS

Section "InputDevice"
  Identifier "*idevname***"**
  Driver "penmount"
  Option "Device" "*devpath***"**
  ...
EndSection

DESCRIPTION

penmount is an XFree86 input driver for PenMount devices...

The **penmount** driver functions as a pointer input device, and may be used as the X server's core pointer. THIS MAN PAGE NEEDS TO BE FILLED IN.

SUPPORTED HARDWARE

What is supported...

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details and for options that can be used with all input drivers. This section only covers configuration details specific to this driver.

Config details...

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7).

AUTHORS

Authors include...

NAME

tek4957 − Tektronix 4957 input driver

SYNOPSIS

Section "InputDevice"
Identifier "*idevname***"**
Driver "tek4957"
Option "Device" "*devpath***"**
...
EndSection

DESCRIPTION

tek4957 is an XFree86 input driver for the Tektronix 4957 tablet.

The **tek4957** driver functions as a pointer input device, and may be used as the X server's core pointer.

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details and for options that can be used with all input drivers. This section only covers configuration details specific to this driver.

Option *"Device" "devpath"*
sets the path to the special file which represents serial line where the tablet is plugged, for example /dev/ttyS0. **This option is mandatory.**

Option *"DeviceName" "name"*
sets the name of the X device.

Option *"Speed" "number"*
sets the sampling rate, from 1 to 6. Default is 6, maximum speed.

Option *"Resolution" "number"*
sets the resolution.
0 : 2340 dots : 1/200 inch
1 : 2972 dots : 1/10 mm
2 : 11700 dots : 1/1000 inch
3 : 11887 dots : 1/40 mm
4 : 5850 dots : 1/500 inch
5 : 5944 dots : 1/20 mm : **default**
6 : 4680 dots : 1/400 inch
7 : 1170 dots : 1/100 inch
8 : 12 dots : 1 inch
9 : 24 dots : 1/2 inch

Option *"TopX" "number"*
X coordinate of the top corner of the active zone. (Default = 0)

Option *"TopY" "number"*
Y coordinate of the top corner of the active zone. (Default = 0)

Option *"BottomX" "Inumber"*
X coordinate of the bottom corner of the active zone. (Default = full scale)

Option *"BottomY" "number"*
Y coordinate of the bottom corner of the active zone. (Default = full scale)

BUGS / LIMITATIONS

Currently, only "Absolute" mode is supported (Sorry)

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7).

AUTHORS

Olivier DANET <odanet@caramail.com>

NAME

UR-98 − UR98 (TR88L803) head tracker driver

SYNOPSIS

Section "InputDevice"
Identifier "*idevname***"**
Driver "UR-98"
Option "Device" "*devpath***"**
...
EndSection

DESCRIPTION

UR-98 is an XFree86 input driver for the Union Reality UR-F98 headtracker.

The **UR-98** driver functions as a pointer input device, and can be used either as an additional input device or as the X server's core pointer. The driver provides support for the three axes, throttle and four buttons of the controller. If mapped as the core pointer the headtracker provides headtracking to try and place the mouse cursor where you look. As a secondary input device the unit can be used for gaming, for example to provide the look up/down and the turn in quake, and with the Z axis bound to ack/forward to provide movement control.

The default mapping maps left-right movement to X, up-down movement to Y and near/far movement to the Z axis. The throttle is mapped as the fourth axis by default but can also be mapped as button 5.

For use in "head only" mode the Z axis can be mapped as a button. This allows the user to select objects with head/neck movement alone but takes some practice to use well.

SUPPORTED HARDWARE

Union Reality UR-98. While this is a joystick driver the behaviour is absolute so this driver is not useful for true joystick interfaces.

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details and for options that can be used with all input drivers. This section only covers configuration details specific to this driver.

The following driver **options** are supported

Option "MinX" "*integer***"**
Set the left hand X value from the headgear, for calibration.

Option "MaxX" "*integer***"**
Set the right hand X value from the headgear, for calibration.

Option "MinY" "*integer***"**
Set the top Y value from the headgear, for calibration.

Option "MaxY" "*integer***"**
Set the bottom Y value from the headgear, for calibration.

Option "MinZ" "*integer***"**
Set the nearest Z value from the headgear, for calibration.

Option "MaxZ" "*integer***"**
Set the furthest Z value from the headgear, for calibration.

Option "MinT" "*integer***"**
Set the low throttle value from the headgear, for calibration.

Option "MaxT" "*integer***"**
Set the high throttle value from the headgear, for calibration.

Option "Screen" "*integer***"**
The screen to attach to the headgear when running with multiple screens. The default is screen 0.

Option "Device" "*string***"**
The joystick port that is attached to the headgear interface. This is usually /dev/input/js0. The digital port is not supported due to lack of documentation.

Option "DeviceName" "*string***"**
Set the X11 device name for the headgear. This defaults to HEAD.

Option "PortraitMode" "*string***"**
Set the display orientation. The default is "landscape" but you can rotate the screen clockwise ("portrait") or anticlockwise ("portraitCCW").

Option "SwapXY" "*boolean***"**
Swap the X and Y values on the display. The default is false.

Option "Button5" "*boolean***"**
Map the throttle as a button instead of axis 4. For some gaming applications this can be more useful. The default is to map the throttle as axis 4.

Option "HeadButton" "*boolean***"**
Map the Z axis as button 1. This defaults to false.

Option "HeadThresh" "*boolean***"**
Set the distance that is held to be mouse down.

Option "HeadLock" "*boolean***"**
Set the range of depth around the mouse down point where mouse x and y movement is locked out. Set to zero to disable.

BUGS

The "HeadButton" option is currently not implemented.

The hardware or kernel driver has some idiosyncracies. Notably on kernel initialization the interface occasionally gets into a state where the readings rapidly cycle left-right-left-right or top-bottom-top-bottom. In those cases it seems to be neccessary to unload the driver, unplug, replug and reload the joystick drivers. Once it initializes sanely it remains sane.

If the device refuses to work check the gray/black cables are plugged into the right ports on the unit. Be careful about this as crossing the cables can lead to the device failing with a nasty burning electronics smell. The author writes from direct experience.

This driver is currently Linux specific.

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7).

AUTHORS

Authors include...
Alan Cox

NAME

void − null input driver

SYNOPSIS

Section "InputDevice"
Identifier "*idevname***"**
Driver "void"
...
EndSection

DESCRIPTION

void is an dummy/null XFree86 input driver. It doesn't connect to any physical device, and it never delivers any events. It functions as both a pointer and keyboard device, and may be used as X server's core pointer and/or core keyboard. It's purpose is to allow the X server to operate without a core pointer and/or core keyboard.

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details and for options that can be used with all input drivers. This driver doesn't have any configuration options in addition to those.

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7).

AUTHORS

Authors include...

NAME

wacom − Wacom input driver

SYNOPSIS

Section "InputDevice"
Identifier "*idevname***"**
Driver "wacom"
Option "Device" "*devpath***"**
...
EndSection

DESCRIPTION

wacom is an XFree86 input driver for Wacom devices.

The **wacom** driver functions as a pointer input device, and may be used as the X server's core pointer.

SUPPORTED HARDWARE

This driver supports the Wacom IV and Wacom V protocols. Preliminary support is available for USB devices on some Linux platforms.

CONFIGURATION DETAILS

Please refer to XF86Config(5) for general configuration details and for options that can be used with all input drivers. This section only covers configuration details specific to this driver.

Multiple instances of the Wacom devices can cohabit. It can be useful to define multiple devices with different active zones. Each device supports the following entries:

Option *"Type" "stylus"|"eraser"|"cursor"*
sets the type of tool the device represent. This option is mandatory.

Option *"Device" "path"*
sets the path to the special file which represents serial line where the tablet is plugged. You have to specify it for each subsection with the same value if you want to have multiple devices with the same tablet. This option is mandatory.

Option *"USB" "on"*
tells the driver to dialog with the tablet the USB way. This option is only available on some Linux platforms.

Option *"DeviceName" "name"*
sets the name of the X device.

Option *"Suppress" "Inumber"*
sets the position increment under which not to transmit coordinates. This entry must be specified only in the first Wacom subsection if you have multiple devices for one tablet. If you don't specify this entry, the default value is computed to

Option *"Mode" "Relative"|"Absolute"*
sets the mode of the device.

Option *"Tilt" "on"*
enables tilt report if your tablet supports it (ROM version 1.4 and above). If this is enabled, multiple devices at the same time will not be reported.

Option *"HistorySize" "number"*
sets the motion history size. By default the value is zero.

Option *"AlwaysCore" "on"*
enables the sharing of the core pointer. When this feature is enabled, the device will take control of the core pointer (and thus will emit core events) and at the same time will be able, when asked so, to report extended events. You can use the last available integer feedback to control this feature. When the value of the feedback is zero, the feature is disabled. The feature is enabled for any other value.

Option *"TopX" "number"*
X coordinate of the top corner of the active zone.

Option *"TopY" "number"*
Y coordinate of the top corner of the active zone.

Option *"BottomX" "Inumber"*
X coordinate of the bottom corner of the active zone.

Option *"BottomY" "number"*
Y coordinate of the bottom corner of the active zone.

Option *"KeepShape" "on"*
When this option is enabled, the active zone begins according to TopX and TopY. The bottom corner is adjusted to keep the ratio width/height of the active zone the same as the screen while maximizing the area described by TopX, TopY, BottomX, BottomY.

Option *"DebugLevel" number*
sets the level of debugging info reported.

Option *"BaudRate" "38400"***,** *"19200"* **or** *"9600"* **(default)**
changes the serial link speed. This option is only available for wacom V models (Intuos).

Option *"Serial" "number"*
sets the serial number associated with the physical device. This allows to have multiple devices of the same type (i.e. multiple pens). This option is only available on wacom V devices (Intuos). To see which serial number belongs to a device, you have to set the DebugLevel to 6 and watch the output of the X server.

Option *"Threshold" "number"*
sets the pressure threshold used to generate a button 1 events of stylus devices for some models of tablets (Intuos and Graphire).

SEE ALSO

XFree86(1), XF86Config(5), xf86config(1), Xserver(1), X(7).

AUTHORS

Frederic Lepied <lepied@xfree86.org>

NAME

XDarwin − X window system server for Darwin operating system

SYNOPSIS

XDarwin [options] ...

DESCRIPTION

XDarwin is the X window server for Mac OS X and the Darwin operating system provided by the XFree86 Project. This version of *XDarwin* can only be started from the Darwin text console. The Mac OS X Aqua GUI, if present, must be shut down. *XDarwin* uses IOKit services to access the display framebuffer, mouse and keyboard and to provide a layer of hardware abstraction. *XDarwin* will normally be started by the *xdm(1)* display manager or by a script that runs the program *xinit(1)*.

OPTIONS

In addition to the normal server options described in the *Xserver(1)* manual page, *XDarwin* accepts the following command line switches:

−fakebuttons
: Emulates a 3 button mouse using modifier keys. By default, the Command modifier is used to emulate button 2 and Option is used for button 3. Thus, clicking the first mouse button while holding down Command will act like clicking button 2. Holding down Option will simulate button 3.

−nofakebuttons
: Do not emulate a 3 button mouse. This is the default.

−fakemouse2 *modifiers*
: Change the modifier keys used to emulate the second mouse button. By default, Command is used to emulate the second button. Any combination of the following modifier names may be used: Shift, Option, Control, Command, Fn. For example, **−fakemouse2 "Option,Shift"** will set holding Option, Shift and clicking on button one as equivalent to clicking the second mouse button.

−fakemouse3 *modifiers*
: Change the modifier keys used to emulate the third mouse button. By default, Option is used to emulate the third button. Any combination of the following modifier names may be used: Shift, Option, Control, Command, Fn. For example, **−fakemouse3 "Control,Shift"** will set holding Control, Shift and clicking on button one as equivalent to clicking the third mouse button.

−keymap *file*
: On startup *XDarwin* translates a Darwin keymapping into an X keymap. The default is to read this keymapping from USA.keymapping. With this option the keymapping will be read from *file* instead. If the file's path is not specified, it will be searched for in Library/Keyboards/ underneath the following directories (in order): ~, /, /Network, /System.

−nokeymap
: On startup *XDarwin* translates a Darwin keymapping into an X keymap. With this option XDarwin queries the kernel for the current keymapping instead of reading it from a file. This will often fail on newer kernels.

−size *width height*
: Sets the screen resolution for the X server to use.

−depth *depth*
: Specifies the color bit depth to use. Currently only 8, 15, and 24 color bits per pixel are supported.

−refresh *rate*
: Gives the refresh rate to use in Hz. For LCD displays this should be 0.

−showconfig
Print out the server version and patchlevel.

−version
Same as **−showconfig**.

SEE ALSO

X(7), XFree86(1), Xserver(1), xdm(1), xinit(1)

BUGS

XDarwin and this man page still have many limitations. Some of the more obvious ones are:
- The display mode cannot be changed once the X server has started.
- A screen saver is not supported.

AUTHORS

XFree86 was originally ported to Mac OS X Server by John Carmack. Dave Zarzycki used this as the basis of his port of XFree86 4.0 to Darwin 1.0. Torrey T. Lyons improved and integrated this code into the XFree86 Project's mainline for the 4.0.2 release.

The following members of the XonX Team contributed to the following releases (in alphabetical order):

XFree86 4.1.0:
Rob Braun - Darwin x86 support
Torrey T. Lyons - Project Lead
Andreas Monitzer - Cocoa version of XDarwin front end
Gregory Robert Parker - Original Quartz implementation
Christoph Pfisterer - Dynamic shared X libraries
Toshimitsu Tanaka - Japanese localization

XFree86 4.2.0:
Rob Braun - Darwin x86 support
Pablo Di Noto - Spanish localization
Paul Edens - Dutch localization
Kyunghwan Kim - Korean localization
Mario Klebsch - Non-US keyboard support
Torrey T. Lyons - Project Lead
Andreas Monitzer - German localization
Patrik Montgomery - Swedish localization
Greg Parker - Rootless support
Toshimitsu Tanaka - Japanese localization
Olivier Verdier - French localization

NAME

dumpkeymap − Dianostic dump of a .keymapping file

SYNOPSIS

dumpkeymap [*options*] [-] [*file*...]

DESCRIPTION

dumpkeymap prints a textual representation of each Apple/NeXT *.keymapping* file mentioned on the command-line. If no files are mentioned and if the local machine is an Apple or NeXT installation, then the key mapping currently in use by the WindowServer and the AppKit is printed instead.

OPTIONS

−h −−help
Display general program instructions and option summary.

−k −−help−keymapping
Display a detailed description of the internal layout of a *.keymapping* file. This is the same information as that presented in the *Key Mapping Description* section of this document.

−o −−help−output
Display an explanation of the output generated by *dumpkeymap* when dissecting a *.keymapping* file. This is the same information as that presented in the *Output Description* section of this document.

−f −−help−files
Display a summary of the various files and directories which are related to key mappings. This is the same information as that presented in the *Files* section of this document.

−d −−help−diagnostics
Display a list of the various diagnostic messages which may be emitted by *dumpkeymap.* This is the same information as that presented in the *Diagnostics* section of this document.

−v −−version
Display the *dumpkeymap* version number and warranty information.

− −− Inhibit processing of options at this point in the argument list. An occurrence of '−' or '−−' in the argument list causes all following arguments to be treated as file names even if an argument begins with a '−' character.

KEY MAPPING DESCRIPTION

The following sections describe, in complete detail, the format of a raw key mapping resource, as well as the format of the *.keymapping* file which encapsulates one or more raw mappings.

Types and Data

The following type definitions are employed throughout this discussion:

```
typedef unsigned char byte;
typedef unsigned short word;
typedef unsigned long dword;
```

Additionally, the type definition '*number*' is used generically to indicate a numeric value. The actual size of the '*number*' type may be one or two bytes depending upon how the data is stored in the key map. Although most key maps use byte-sized numeric values, word-sized values are also allowed.

Multi-byte values in a key mapping file are stored in big-endian byte order.

Key Mapping File and Device Mapping

A key mapping file begins with a magic-number and continues with a variable number of device-specific key mappings.

```
struct KeyMappingFile {
    char magic_number[4];       // 'KYM1'
    DeviceMapping maps[...];    // Variable number of maps
};
```

```
struct DeviceMapping {
    dword interface;      // Interface type
    dword handler_id;     // Interface subtype
    dword map_size;       // Byte count of 'map' (below)
    KeyMapping map;
};
```

The value of 'interface' represents a family of keyboard device types (such as Intel PC, ADB, NeXT, Sun Type5, etc.), and is generally specified as one of the constant values NX_EVS_DEVICE_INTERFACE_ADB, NX_EVS_DEVICE_INTERFACE_ACE, etc., which are are defined in IOHIDTypes.h on MacOS/X and Darwin, and in ev_types.h on MacOS/X Server, OpenStep, and NextStep.

The value of 'handler_id' represents a specific keyboard layout within the much broader 'interface' family. For instance, for a 101-key Intel PC keyboard (of type NX_EVS_DEVICE_INTERFACE_ACE) the 'handler_id' is '0', whereas for a 102-key keyboard it is '1'.

Together, 'interface' and 'handler_id' identify the exact keyboard hardware to which this mapping applies. Programs which display a visual representation of a keyboard layout, match 'interface' and 'handler_id' from the *.keymapping* file against the 'interface' and 'handler_id' values found in each *.keyboard* file.

Key Mapping

A key mapping completely defines the relationship of all scan codes with their associated functionality. A *KeyMapping* structure is embedded within the *DeviceMapping* structure in a *KeyMappingFile*. The key mapping currently in use by the WindowServer and AppKit is also represented by a *KeyMapping* structure, and can be referred to directly by calling NXGetKeyMapping() and accessing the 'mapping' data member of the returned *NXKeyMapping* structure.

```
struct KeyMapping {
    word number_size;               // 0=1 byte, non-zero=2 bytes
    number num_modifier_groups;     // Modifier groups
    ModifierGroup modifier_groups[...];
    number num_scan_codes;          // Scan groups
    ScanGroup scan_table[...];
    number num_sequence_lists;      // Sequence lists
    Sequence sequence_lists[...];
    number num_special_keys;        // Special keys
    SpecialKey special_key[...];
};
```

The 'number_size' flag determines the size, in bytes, of all remaining numeric values (denoted by the type definition '*number*') within the key mapping. If its value is zero, then numbers are represented by a single byte. If it is non-zero, then numbers are represented by a word (two bytes).

Modifier Group

A modifier group defines all scan codes which map to a particular type of modifier, such as *shift*, *control*, etc.

```
enum Modifier {
    ALPHALOCK = 0,
    SHIFT,
    CONTROL,
    ALTERNATE,
    COMMAND,
    KEYPAD,
    HELP
};
```

```
struct ModifierGroup {
    number modifier;            // A Modifier constant
    number num_scan_codes;
```

```
    number scan_codes[...];     // Variable number of scan codes
};
```

The scan_codes[] array contains a list of all scan codes which map to the specified modifier. The *shift*, *command*, and *alternate* modifiers are frequently mapped to two different scan codes, apiece, since these modifiers often appear on both the left and right sides of the keyboard.

Scan Group

There is one *ScanGroup* for each scan code generated by the given keyboard. This number is given by KeyMapping::num_scan_codes. The first scan group represents hardware scan code 0, the second represents scan code 1, etc.

```
enum ModifierMask {
    ALPHALOCK_MASK          = 1 << 0,
    SHIFT_MASK              = 1 << 1,
    CONTROL_MASK            = 1 << 2,
    ALTERNATE_MASK          = 1 << 3,
    CARRIAGE_RETURN_MASK    = 1 << 4
};
#define NOT_BOUND 0xff

struct ScanGroup {
    number mask;
    Character characters[...];
};
```

For each scan code, ‘mask’ defines which modifier combinations generate characters. If ‘mask’ is NOT_BOUND (0xff) then then this scan code does not generate any characters ever, and its characters[] array is zero length. Otherwise, the characters[] array contains one *Character* record for each modifier combination.

The number of records in characters[] is determined by computing (1 << bits_set_in_mask). In other words, if mask is zero, then zero bits are set, so characters[] contains only one record. If ‘mask’ is (SHIFT_MASK | CONTROL_MASK), then two bits are set, so characters[] contains four records.

The first record always represents the character which is generated by that key when no modifiers are active. The remaining records represent characters generated by the various modifier combinations. Using the example with the *shift* and *control* masks set, record two would represent the character with the *shift* modifier active; record three, the *control* modifier active; and record four, both the *shift* and *control* modifiers active.

As a special case, ALPHALOCK_MASK implies SHIFT_MASK, though only ALPHALOCK_MASK appears in ‘mask’. In this case the same character is generated for both the *shift* and *alpha-lock* modifiers, but only needs to appear once in the characters[] array.

CARRIAGE_RETURN_MASK does not actually refer to a modifier key. Instead, it is used to distinguish the scan code which is given the special pseudo-designation of *carriage return* key. Typically, this mask appears solo in a *ScanGroup* record and only the two *Character* records for control-M and control-C follow. This flag may be a throwback to an earlier time or may be specially interpreted by the low-level keyboard driver, but its purpose is otherwise enigmatic.

Character

Each *Character* record indicates the character generated when this key is pressed, as well as the character set which contains the character. Well known character sets are ‘ASCII’ and ‘Symbol’. The character set can also be one of the meta values FUNCTION_KEY or KEY_SEQUENCE. If it is FUNCTION_KEY then ‘char_code’ represents a generally well-known function key such as those enumerated by *FunctionKey*. If the character set is KEY_SEQUENCE then ‘char_code’ represents is a zero-base index into KeyMapping::sequence_lists[].

```
enum CharacterSet {
    ASCII               = 0x00,
```

```
    SYMBOL          = 0x01,
    ...
    FUNCTION_KEY    = 0xfe,
    KEY_SEQUENCE    = 0xff
};

struct Character {
    number set;         // CharacterSet of generated character
    number char_code;   // Actual character generated
};

enum FunctionKey {
    F1 = 0x20, F2, F3, F4, F5, F6, F7, F8, F9, F10, F11, F12,
    INSERT, DELETE, HOME, END, PAGE_UP, PAGE_DOWN, PRINT_SCREEN,
    SCROLL_LOCK, PAUSE, SYS_REQUEST, BREAK, RESET, STOP, MENU,
    USER, SYSTEM, PRINT, CLEAR_LINE, CLEAR_DISPLAY, INSERT_LINE,
    DELETE_LINE, INSERT_CHAR, DELETE_CHAR, PREV, NEXT, SELECT
};
```

Sequence

When Character::set contains the meta value KEY_SEQUENCE, the scan code is bound to a sequence of keys rather than a single character. A sequence is a series of modifiers and characters which are automatically generated when the associated key is depressed.

```
#define MODIFIER_KEY 0xff

struct Sequence {
    number num_chars;
    Character characters[...];
};
```

Each generated *Character* is represented as previously described, with the exception that MODIFIER_KEY may appear in place of KEY_SEQUENCE. When the value of Character::set is MODIFIER_KEY then Character::char_code represents a modifier key rather than an actual character. If the modifier represented by ‘char_code’ is non-zero, then it indicates that the associated modifier key has been depressed. In this case, the value is one of the constants enumerated by *Modifier* (SHIFT, CONTROL, ALTERNATE, etc.). If the value is zero then it means that the modifier keys have been released.

Special Key

A special key is one which is scanned directly by the Mach kernel rather than by the WindowServer. In general, events are not generated for special keys.

```
enum SpecialKeyType {
    VOLUME_UP = 0,
    VOLUME_DOWN,
    BRIGHTNESS_UP,
    BRIGHTNESS_DOWN,
    ALPHA_LOCK,
    HELP,
    POWER,
    SECONDARY_ARROW_UP,
    SECONDARY_ARROW_DOWN
};

struct SpecialKey {
    number type;        // A SpecialKeyType constant
    number scan_code;   // Actual scan code
};
```

OUTPUT

What follows is an explanation and description of the various pieces of information emitted by *dump-keymap.*

For a more thorough discussion of any particular piece of information described here, refer to the detailed description of the internal layout of a key mapping provided by the *Key Mapping Description* section above.

Conventions

Depending upon context, some numeric values are displayed in decimal notation, whereas others are displayed in hexadecimal notation. Hexadecimal numbers are denoted by a ‘0x’ prefix (for instance, ‘0x7b’), except when explicitly noted otherwise.

Key Mapping Source

The first piece of information presented about a particular key mapping is the source from which the data was gleaned. For a *.keymapping* file, the title ‘KEYMAP FILE’ is emitted along with the path and name of the file in question. For the key mapping currently in use by the WindowServer and AppKit, the title ‘ACTIVE KEYMAP’ is emitted instead.

Device Information

Each *.keymapping* file may contain one or more raw key mappings. For example, a file which maps keys to a Dvorak-style layout might contain raw mappings for Intel PC, ADB, NeXT, and Sun Type5 keyboards.

For each raw mapping, the following information is emitted:

- The title ‘KEYMAP’ along with the mapping’s relative position in the *.keymapping* file.
- The ‘interface’ identifier.
- The ‘handler_id’ sub-identifier.
- The size of the raw mapping resource counted in bytes.

The ‘interface’ and ‘handler_id’ values, taken together, define a specific keyboard device. A *.keyboard* file, which describes the visual layout of a keyboard, also contains ‘interface’ and ‘handler_id’ identifiers. The *.keyboard* file corresponding to a particular key mapping can be found by matching the ‘interface’ and ‘handler_id’ values from each resource.

Modifiers

Each mapping may contain zero or more modifier records which associate hardware scan codes with modifier descriptions such as *shift, control, alternate,* etc. The title ‘MODIFIERS’ is printed along with the count of modifier records which follow. For each modifier record, the modifier’s name is printed along with a list of scan codes, in hexadecimal format, which generate that modifier value. For example:

```
MODIFIERS [4]
alternate: 0x1d 0x60
control: 0x3a
keypad: 0x52 0x53 ... 0x63 0x62
shift: 0x2a 0x36
```

Characters

Each mapping may contain zero or more character records which associate hardware scan codes with the actual characters generated by those scan codes in the presence or absence of various modifier combinations. The title ‘CHARACTERS’ is printed along with the count of character records which follow. Here is a highly abbreviated example:

```
CHARACTERS [9]
scan 0x00: -AC-L  "a" "A" "^A" "^A" ca c7 "^A" "^A"
scan 0x07: -AC-L  "x" "X" "^X" "^X" 01/b4 01/ce "^X" "^X"
scan 0x0a: ---S-  "<" ">"
scan 0x13: -ACS-  "2" "@" "^@" "^@" b2 b3 "^@" "^@"
scan 0x24: R----  "^M" "^C"
```

```
scan 0x3e: -----  [F4]
scan 0x4a: -----  [page up]
scan 0x60: -----  {seq#3}
scan 0x68: not-bound
```

For each record, the hexadecimal value of the hardware scan code is printed, followed by a list of modifier flag combinations and the actual characters generated by this scan code with and without modifiers applied.

The modifier flags field is composed of a combination of single letter representations of the various modifier types. The letters stand for:

L – alpha-lock
S – shift
C – control
A – alternate
R – carriage-return

As a special case, the *alpha-lock* flag also implies the *shift* flag, so these two flags never appear together in the same record.

The combination of modifier flags determines the meaning and number of fields which follow. The first field after the modifier flags always represents the character that will be generated if no modifier keys are depressed. The remaining fields represent characters generated by the various modifier combinations. The order of the fields follows this general pattern:

- The character generated by this scan code when no modifiers are in effect is listed first.
- If the 'L' or 'S' flag is active, then the shifted character generated by this scan code is listed next.
- If the 'C' flag is active, then the control-character generated by this scan code is listed next. Furthermore, if the 'L' or 'S' flag is also active, then the shifted control-character is listed after that.
- If the 'A' flag is active, then the alternate-character generated by this scan code is listed next. Furthermore, if the 'L' or 'S' flag is active, then the shifted alternate-character is listed after that. If the 'C' flag is also active, then the alternate-control-character is listed next. Finally, if the 'C' and 'L' or 'C' and 'S' flags are also active, then the shifted alternate-control-character is listed.

The 'R' flag does not actually refer to a modifier key. Instead, it is used to distinguish the scan code which is given the special pseudo-designation of *carriage return* key. Typically, this mask appears solo and only the two fields for control-M and control-C follow. This flag may be a throwback to an earlier time or may be specially interpreted by the low-level keyboard driver, but its purpose is otherwise enigmatic.

Recalling the example from above, the following fields can be identified:

```
scan 0x00: -AC-L  "a" "A" "^A" "^A" ca c7 "^A" "^A"
```

- Lower-case 'a' is generated when no modifiers are active.
- Upper-case 'A' is generated when *shift* or *alpha-lock* are active.
- Control-A is generated when *control* is active.
- Control-A is generated when *control* and *shift* are active.
- The character represented by the hexadecimal code 0xca is generated when *alternate* is active.
- The character represented by 0xc7 is generated when *alternate* and *shift* (or *alpha-lock*) are active.
- Control-A is generated when *alternate* and *control* are active.
- Control-A is generated when *alternate, control* and *shift* (or *alpha-lock*) are active.

The notation used to represent a particular generated character varies.

- Printable ASCII characters are quoted, as in "x" or "X".
- Control-characters are quoted and prefixed with ‘^’, as in "^X".
- Characters with values greater than 127 (0x7f) are displayed as hexadecimal values without the ‘0x’ prefix.
- Characters in a non-ASCII character set (such as ‘Symbol’) are displayed as two hexadecimal numbers separated by a slash, as in ‘01/4a’. The first number is the character set’s identification code (such as ‘01’ for the ‘Symbol’ set), and the second number is the value of the generated character.
- Non-printing special function characters are displayed with the function’s common name enclosed in brackets, as in ‘[page up]’ or ‘[F4]’.
- If the binding represents a key sequence rather than a single character, then the sequence’s identification number is enclosed in braces, as in ‘{seq#3}’.

Recalling a few examples from above, the following interpretations can be made:

```
scan 0x07: -AC-L  "x" "X" "^X" "^X" 01/b4 01/ce "^X" "^X"
scan 0x3e: -----  [F4]
scan 0x4a: -----  [page up]
scan 0x60: -----  {seq#3}
```

- "x" and "X" are printable ASCII characters.
- "^X" is a control-character.
- ‘01/b4’ and ‘01/ce’ represent the character codes 0xb4 and 0xce in the ‘Symbol’ character set.
- Scan code 0x3e generates function-key ‘F4’, and scan code 0x4a generates function-key ‘page up’.
- Scan code 0x60 is bound to key sequence #3.

Finally, if a scan code is not bound to any characters, then it is annotated with the label ‘not-bound’, as with example scan code 0x68 from above.

Sequences

A scan code (modified and unmodified) can be bound to a key sequence rather than generating a single character or acting as a modifier. When it is bound to a key sequence, a series of character invocations and modifier actions are automatically generated rather than a single keystroke.

Each mapping may contain zero or more key sequence records. The title ‘SEQUENCES’ is printed along with the count of sequence records which follow. For example:

```
SEQUENCES [3]
sequence 0: "f" "o" "o"
sequence 1: {alternate} "b" "a" "r" {unmodify}
sequence 2: [home] "b" "a" "z"
```

The notation used to represent the sequence of generated characters is identical to the notation already described in the *Characters* section above, with the exception that modifier actions may be interposed between generated characters. Such modifier actions are represented by the modifier’s name enclosed in braces. The special name ‘{unmodify}’ indicates the release of the modifier keys.

Thus, the sequences in the above example can be interpreted as follows:

- Sequence #0 generates ‘foo’.
- Sequence #1 invokes the *alternate* modifier, generates ‘bar’, and then releases *alternate.*
- Sequence #2 invokes the *home* key and then generates ‘baz’. In a text editor, this would probably result in ‘baz’ being prepended to the line of text on which the cursor resides.

Special Keys

Certain keyboards feature keys which perform some type of special purpose function rather than generating a character or acting as a modifier. For instance, Apple keyboards often contain a *power* key, and NeXT keyboards have historically featured screen brightness and volume control keys.

Each mapping may contain zero or more special-key records which associate hardware scan codes with such special purpose functions. The title 'SPECIALS' is printed along with the count of records which follow. For each record, the special function's name is printed along with a list of scan codes, in hexadecimal format, which are bound to that function. For example:

```
SPECIALS [6]
alpha-lock: 0x39
brightness-down: 0x79
brightness-up: 0x74
power: 0x7f
sound-down: 0x77
sound-up: 0x73
```

FILES

*.keymapping
: A key mapping file which precisely defines the relationship of all hardware-specific keyboard scan-codes with their associated functionality.

*.keyboard
: A file describing the physical layout of keys on a particular type of keyboard. Each 'key' token in this file defines the position and shape of the key on the keyboard, as well as the associated scan code which that key generates. A *.keymapping* file, on the other hand, defines the characters which are generated by a particular scan code depending upon the state of the various modifier keys (such as *shift, control,* etc.). The 'interface' and 'handler_id' values from a *.keymapping* file are matched against those in each *.keyboard* file in order to associate a particular *.keyboard* file with a key mapping. Various GUI programs use the *.keyboard* file to display a visual representation of a keyboard for the user. Since these files are just plain text, they can be easily viewed and interpreted without the aid of a specialized program, thus *dumpkeymap* leaves these files alone.

/System/Library/Keyboards
/Network/Library/Keyboards
/Local/Library/Keyboards
/Library/Keyboards
: Repositories for *.keymapping* and *.keyboard* files for MacOS/X, Darwin, and MacOS/X Server.

/NextLibrary/Keyboards
/LocalLibrary/Keyboards
: Repositories for *.keymapping* and *.keyboard* files for OpenStep and NextStep.

$(HOME)/Library/Keyboards
: Repository for personal *.keymapping* and *.keyboard* files.

DIGANOSTICS

The following diagnostic messages may be issued to the standard error stream.

Unrecognized option.
: An unrecognized option was specified on the command-line. Invoke *dumpkeymap* with the **−−help** option to view a list of valid options.

Insufficient data in keymapping data stream.
: The key mapping file or data stream is corrupt. Either the file has been incorrectly truncated or a field, such as those which indicates the number of variable records which follow, contains a corrupt value.

The following diagnostic messages have significance only when trying to print *.keymapping* files mentioned on the command-line.

Bad magic number.
The mentioned file is not a *.keymapping* file. The file's content does not start with the string 'KYM1'.

Unable to open key mapping file.
The call to fopen() failed; probably because the specified path is invalid or *dumpkeymap* does not have permission to read the file.

Unable to determine key mapping file size.
The call to fstat() failed, thus memory can not be allocated for loading the file.

Unable to read key mapping file.
The call to fread() failed.

The following diagnostic messages have significance only when trying to print the currently active key mapping when no *.keymapping* files have been mentioned on the command-line.

Unable to open event status driver.
The call to NXOpenEventStatus() failed.

Bad key mapping length.
The call to NXKeyMappingLength() returned a bogus value.

Unable to get current key mapping.
The call to NXGetKeyMapping() failed.

The following diagnostic messages have significance only when using *dumpkeymap* on a non-Apple/NeXT platform.

Must specify at least one .keymapping file.
No *.keymapping* files were mentioned on the command-line. On non-Apple/NeXT platforms, there is no concept of a currently active *.keymapping* file, so at least one file must be mentioned on the command-line.

AUTHOR

Eric Sunshine <sunshine@sunshineco.com> wrote *dumpkeymap* and this document, the *dumpkeymap user's manual.* Both *dumpkeymap* and this document are copyright ©1999,2000 by Eric Sunshine <sunshine@sunshineco.com>. All rights reserved.

The implementation of *dumpkeymap* is based upon information gathered on September 3, 1997 by Eric Sunshine <sunshine@sunshineco.com> and Paul S. McCarthy <zarnuk@zarnuk.com> during an effort to reverse engineer the format of the NeXT *.keymapping* file.

NAME

Xkdrive − tiny X server

SYNOPSIS

Xvesa [*:display*] [*option*...]

Xfbdev [*:display*] [*option*...]

Xigs [*:display*] [*option*...]

Xtrident [*:display*] [*option*...]

Xsis530 [*:display*] [*option*...]

Xtrio [*:display*] [*option*...]

Xitsy [*:display*] [*option*...]

DESCRIPTION

Xkdrive is a family of X servers designed to be particularly small. This manual page describes the common functionality of the **Xkdrive** servers; for information on a specific X server, please refer to the relevant manual page.

OPTIONS

In addition to the standard options accepted by all X servers (see Xserver(1)), all the **Xkdrive** servers accept the following options:

-card *pcmcia*
use pcmcia card as additional screen.

-dumb disable hardware acceleration.

-origin *X*,*Y*
Locates the next screen in the Xinerama virtual screen.

-screen *width***x***height*[**x***depth*[**x***freq*]][@*rotation*]
use a screen of the specified *width*, *height*, screen *depth*, *frequency*, and *rotation* (0, 90, 180 and 270 are legal values).

-softCursor
disable the hardware cursor.

-videoTest
start the server, pause momentarily, and exit.

-zaphod disable switching screens by moving the pointer across a screen boundary.

-2button
enable emulation of a middle mouse button by chording.

-3button
disable emulation of a middle mouse button by chording.

SEE ALSO

X(7), Xserver(1), xdm(1), xinit(1), Xvesa(1), Xfbdev(1).

AUTHORS

The Xkdrive common core was written by Keith Packard, based on XFree86 which, in turn, is based on the Sample Implementation of X.

NAME

Xvesa − VESA Bios Extensions tiny X server

SYNOPSIS

Xvesa [*:display*] [*option*...]

DESCRIPTION

Xvesa is a generic X server for Linux on the x86 platform. **Xvesa** doesn't know about any particular hardware, and sets the video mode by running the video BIOS in VM86 mode. **Xvesa** can use both standard VGA BIOS modes and any modes advertised by a VESA BIOS if available.

Xvesa runs untrusted code with full privileges, and is therefore a fairly insecure X server. **Run at your own risk.**

OPTIONS

In addition to the normal KDrive server's options (see Xkdrive(1)), **Xvesa** accepts the following command line switches:

-mode *n* specifies the VESA video mode to use. If mode *n* is not supported by your BIOS and hardware, **Xvesa** will fail, hang your system, damage your hardware, or cause a global thermonuclear war; you are on your own. This option overrides any **-screen** options.

-listmodes
list all supported video modes. If **-force** was specified before **-listmodes**, lists all the modes that your BIOS claims to support, even those that the **Xvesa** server won't be able to use.

-force disable some sanity checks and use the specified mode even if the BIOS claims not to support it.

-shadow
use a shadow framebuffer even if it is not strictly necessary. This may dramatically improve performance on some hardware.

-nolinear
don't use a linear framebuffer even if one is available. You don't want to use this option.

-swaprgb
pass RGB values in the order that works on broken BIOSes. Use this if the colours are wrong in PseudoColor and 16 colour modes.

-map-holes
use a contiguous (hole-less) memory map. This fixes a segmentation violation with some rare BIOSes that violate the VESA specification, but may cause slightly higher memory usage on systems that overcommit memory.

-verbose
emit diagnostic messages during BIOS initialization and teardown.

KEYBOARD

Multiple key presses recognized directly by **Xvesa** are:

Ctrl+Alt+Backspace
Immediately kill the server.

Ctrl+Alt+F1...F12
Switch to virtual console 1 through 12.

BUGS

Xvesa opens all IO ports and runs your VESA BIOS, which may be assumed to be buggy. Allowing your users to run **Xvesa** is probably a security hole.

Xvesa records the current BIOS mode when it starts and restores that mode on termination; if the video card has been reprogrammed by another application, the display will almost certainly be trashed. The alternative of saving and restoring the complete video card state has proven unreliable on most video cards.

SEE ALSO

X(7), Xserver(1), Xkdrive(1), xdm(1), xinit(1).

AUTHORS

The VESA driver was written by Juliusz Chroboczek who didn't realise what he was doing until it was too late. Keith Packard then added support for standard VGA BIOS modes and is especially proud of 320x200 16 colour mode.

NAME

Xfbdev − Linux framebuffer device tiny X server

SYNOPSIS

Xfbdev [*:display*] [*option*...]

DESCRIPTION

Xfbdev is a generic X server for Linux. **Xfbdev** doesn't know about any particular hardware, and uses the framebuffer provided by the Linux framebuffer device.

OPTIONS

Xfbdev accepts the common options of the Xkdrive family of servers. Please see Xkdrive(1).

KEYBOARD

To be written.

SEE ALSO

X(7), Xserver(1), Xkdrive(1), xdm(1), xinit(1).

AUTHORS

The **Xfbdev** server was written by Keith Packard.

NAME

XDGA − XFree86 DGA extension client library

SYNOPSIS

#include <X11/extensions/xf86dga.h>

Bool **XDGAQueryExtension**(
Display **dpy*,
int **eventBase*,
int **errorBase*)

Bool **XDGAQueryVersion**(
Display **dpy*,
int **majorVersion*,
int **minorVersion*)

XDGAMode ***XDGAQueryModes**(
Display **dpy*,
int *screen*,
int **num*)

XDGADevice ***XDGASetMode**(
Display **dpy*,
int *screen*,
int *mode*)

Bool **XDGAOpenFramebuffer**(
Display **dpy*,
int *screen*)

void **XDGACloseFramebuffer**(
Display **dpy*,
int *screen*)

void **XDGASetViewport**(
Display **dpy*,
int *screen*,
int *x*,
int *y*,
int *flags*)

void **XDGAInstallColormap**(
Display **dpy*,
int *screen*,
Colormap *cmap*)

Colormap **XDGACreateColormap**(
Display **dpy*,
int *screen*,
XDGADevice **device*,
int *alloc*)

void **XDGASelectInput**(
Display **dpy*,
int *screen*,
long *event_mask*)

void **XDGAFillRectangle**(
Display **dpy*,
int *screen*,
int *x*,
int *y*,

unsigned int *width*,
unsigned int *height*,
unsigned long *color*)

void **XDGACopyArea**(
Display **dpy*,
int *screen*,
int *srcx*,
int *srcy*,
unsigned int *width*,
unsigned int *height*,
int *dstx*,
int *dsty*)

void **XDGACopyTransparentArea**(
Display **dpy*,
int *screen*,
int *srcx*,
int *srcy*,
unsigned int *width*,
unsigned int *height*,
int *dstx*,
int *dsty*,
unsigned long *key*)

int **XDGAGetViewportStatus**(
Display **dpy*,
int *screen*)

void **XDGASync**(
Display **dpy*,
int *screen*)

Bool **XDGASetClientVersion**(
Display **dpy*)

void **XDGAChangePixmapMode**(
Display **dpy*,
int *screen*,
int **x*,
int **y*,
int *mode*)

void **XDGAKeyEventToXKeyEvent**(
XDGAKeyEvent **dk*,
XKeyEvent **xk*)

DESCRIPTION

The **XFree86-DGA** extension is an X server extension for allowing client programs direct access to the video frame buffer. This is a brief description of the programming interface for version 2.0 of the **XFree86-DGA** extension.

XFree86-DGA is not intended as a direct rendering API, but rather, as a mechanism to "get the X Server out of the way" so that some other direct rendering API can have full access to the hardware. With this in mind, DGA does provide clients some direct access to the hardware without requiring a separate rendering API, but this access is limited to direct linear framebuffer access.

Most of the reasons for the **XFree86-DGA** extension's existence are now better served in other ways. Further development of this extension is not expected, and it may be deprecated in a future release. The features that continue to be useful will either be provided through other existing mechanisms, or through an extension that address those needs more specifically. Discussion of these issue is encouraged in the XFree86 developer forum <*devel@xfree86.org*>.

XFree86-DGA is initialized by passing a number corresponding to a valid *XDGAMode* to **XDGASetMode**(). Clients can get a list of valid modes from **XDGAQueryModes**(). Each *XDGAMode* corresponds to a different framebuffer layout.

XDGAQueryModes() returns a pointer to an array of *XDGAMode*s which are valid for the given screen. *num* is the number of elements in the array. The returned array can be freed with XFree(3). The *XDGAMode* structure is as follows:

```
typedef struct {
  int num;
  char *name;
  float verticalRefresh;
  int flags;
  int imageWidth;
  int imageHeight;
  int pixmapWidth;
  int pixmapHeight;
  int bytesPerScanline;
  int byteOrder;
  int depth;
  int bitsPerPixel;
  unsigned long redMask;
  unsigned long greenMask;
  unsigned long blueMask;
  short visualClass;
  int viewportWidth;
  int viewportHeight;
  int xViewportStep;
  int yViewportStep;
  int maxViewportX;
  int maxViewportY;
  int viewportFlags;
  int reserved1;
  int reserved2;
} XDGAMode;
```

num A unique identifying number (*num* > 0) for the mode. This is the number referenced when initializing the mode.

name The name of the corresponding modeline as given in the XF86Config file.

verticalRefresh
The vertical refresh rate for the modeline (in Hz).

flags Any of the following may be OR'd together:

XDGAConcurrentAccess
Indicates that concurrent client/server access to the framebuffer is possible. If this flag is not set it is very important to call **XDGASync**() before directly accessing the framebuffer if a call to **XDGAFillRectangle**(), **XDGACopyArea**() or **XDGACopyTransparentArea**() or any Xlib rendering function has been made prior to such accesses.

XDGASolidFillRect
Indicates that **XDGAFillRectangle**() is supported.

XDGABlitRect
Indicates that **XDGACopyArea**() is supported.

XDGABlitTransRect
Indicates that **XDGACopyTransparentArea**() is supported.

XDGAPixmap
Indicates that a Pixmap will be returned when the mode is initialized. This means that rendering with Xlib is possible for this mode.

XDGAInterlaced

XDGADoublescan
Indicates that the mode is an interlaced or doublescan mode.

imageWidth

imageHeight
The width and height of the framebuffer area accessible by the client. This rectangle is always justified to the upper left-hand corner.

pixmapWidth

pixmapHeight
The width and height of the framebuffer area accessible by Xlib. This rectangle is always justified to the upper left-hand corner. These fields are only valid if the **XDGAPixmap** flag is set in the *flags* field.

bytesPerScanline
The pitch of the framebuffer in bytes.

byteOrder
MSBFirst or **LSBFirst**.

depth The number of bits in each pixel which contain usable data.

bitsPerPixel
The number of bits taken up by each pixel.

redMask

greenMask

blueMask
The RGB masks. These do not apply to color-indexed modes.

visualClass
TrueColor, **PseudoColor**, **DirectColor**, etc.

viewportWidth

viewportHeight
The dimensions of the portion of the framebuffer which will be displayed on the screen.

xViewPortStep

yViewPortStep
The granularity of the x,y viewport positioning possible with the **XDGASetViewport**() function.

maxViewportX

maxViewportY
The maximum x and y positions possible with the **XDGASetViewport**() function.

viewportFlags
Any of the following may be OR'd together

XDGAFlipRetrace
Indicates that the hardware can switch viewports during the vertical retrace.

XDGAFlipImmediate
Indicates that the hardware can switch viewports immediately without waiting for the vertical retrace.

XDGASetMode() initialises the *XDGAMode* corresponding to *num*. To exit DGA mode and return to normal server operation, call **XDGASetMode**() with *num* set to zero. **XDGASetMode**() returns a pointer to an *XDGADevice* if successful. The XDGADevice can be freed with XFree(3). The *XDGADevice* structure is as follows:

```
typedef struct {
  XDGAMode mode;
  unsigned char *data;
  Pixmap pixmap;
} XDGADevice;
```

mode The *XDGAMode* structure, identical to the information returned by **XDGAQueryModes**().

data If direct framebuffer access is desired and possible, this field will contain a pointer to the mapped framebuffer memory. Generally, this field will be zero unless a call to **XDGAOpenFramebuffer**() is made prior to initialization of the mode.

pixmap If the mode supports Xlib rendering as indicated by **XDGAPixmap** in the *flags* field, this will contain a Pixmap handle suitable for passing as the drawable argument to Xlib functions. This field will be zero if Xlib rendering is not supported.

XDGAQueryExtension() checks for the presence of the extension and returns the event and error bases.

XDGAQueryVersion() returns the **XFree86-DGA** major and minor version numbers.

XDGAOpenFramebuffer() maps the framebuffer memory. The client needs sufficient privileges to be able to do this. **XDGAOpenFramebuffer**() should be called prior to initializing a DGA mode if direct framebuffer access is desired for that mode. **XDGAOpenFramebuffer**() does not need to be called if direct framebuffer access is not required. If the framebuffer is opened,

XDGACloseFramebuffer() should be called prior to client exit to unmap the memory.

XDGAChangePixmapMode() can be used to change between two pixmap sizes in cases where a Pixmap is available for Xlib rendering. The following values for the *mode* parameter are available:

XDGAPixmapModeLarge
The pixmap size is defined by the *pixmapWidth* and *pixmapHeight* fields in the *XDGAMode* structure. The *x* and *y* values are ignored in this case.

XDGAPixmapModeSmall
The pixmap size is defined by the *viewportWidth* and *viewportHeight* fields in the *XDGAMode* structure. In this mode, the *x* and *y* values specify where in the framebuffer this pixmap rectangle is located. It may be placed anywhere within the Xlib renderable region described by the *pixmapWidth* and *pixmapHeight* fields in the *XDGAMode*. The *x* and *y* values returned are the resultant location of the pixmap and may be different from the requested x,y location due to platform specific alignment constraints. All Xlib rendering is clipped to this pixmap rectangle.

XDGASetViewport() sets the upper left-hand corner of the rectangle of framebuffer that is to be displayed on the screen. Not all locations may be supported by the hardware and requested locations will be adjusted according to the *xViewPortStep* and *yViewPortStep* fields in the *XDGAMode*.

flags can be **XDGAFlipRetrace** or **XDGAFlipImmediate** to adjust the viewport location at the next vertical retrace or immediately. Values other than the supported values advertised in the mode's *viewportFlags* field will result in hardware-specific default behavior. **XDGAFlipImmediate** will block until the flip is completed. **XDGAFlipRetrace** will generally NOT block so it is necessary to monitor the viewport status with **XDGAGetViewportStatus**(). **XDGAFlipImmediate** requests during pending **XDGAFlipRetrace** requests will be ignored.

XDGAGetViewportStatus() keeps track of the **XDGASetViewport**() requests still pending. The return value of the function will have consecutive bits set (LSB justified), each bit representing a pending viewport change. For example:

```
while(XDGAGetViewportStatus(dpy, screen));
```

waits for all pending viewport changes to finish.

```
while(0x2 & XDGAGetViewportStatus(dpy, screen));
```

waits until all but the last viewport changes have completed.

XDGACreateColormap() is similar to the Xlib function XCreateColormap(3) except that it takes an *XDGADevice* as an argument instead of a Window and Visual. Though XCreateColormap(3) may create usable colormaps in some cases, **XDGACreateColormap**() is the preferred method for creating colormaps in DGA since there may not be an advertised visual compatible with the DGA device.

XDGAInstallColormap() must be used to install colormaps in DGA mode. XInstallColormap(3) will not work.

XDGASelectInput() enables DGA's own event mechanism. This function is similar to XSelectInput(3), and all Xlib Key, Button and Motion masks are supported. The following DGA events are defined:

```
typedef struct {
   int type;             /* ButtonPress or ButtonRelease + the DGA event base*/
   unsigned long serial; /* # or last request processed by the server */
   Display *display;     /* Display the event was read from */
   int screen;           /* The screen number the event came from */
   Time time;            /* milliseconds */
   unsigned int state;   /* key or button mask */
   unsigned int button;  /* detail */
} XDGAButtonEvent;

typedef struct {
   int type;             /* KeyPress or KeyRelease + the DGA event base*/
   unsigned long serial; /* # or last request processed by the server */
   Display *display;     /* Display the event was read from */
   int screen;           /* The screen number the event came from */
   Time time;            /* milliseconds */
   unsigned int state;   /* key or button mask */
   unsigned int keycode; /* detail */
} XDGAKeyEvent;

typedef struct {
   int type;             /* MotionNotify + the DGA event base*/
   unsigned long serial; /* # or last request processed by the server */
   Display *display;     /* Display the event was read from */
   int screen;           /* The screen number the event came from */
   Time time;            /* milliseconds */
   unsigned int state;   /* key or button mask */
   int dx;               /* relative pointer motion */
   int dy;               /* relative pointer motion */
} XDGAMotionEvent;
```

XDGAKeyEventToXKeyEvent() is a helper function to translate *XDGAKeyEvent*s into *XKeyEvent*s suitable for use with XLookupKeysym(3).

XDGAFillRectangle(), **XDGACopyArea**(), and **XDGACopyTransparentArea**() are included with some reservation since DGA is not intended as a rendering API. These are merely convenience routines and are optionally supported. The associated flags will be set in the *XDGAMode*'s *flags* field if these functions are supported. These functions will be no-ops otherwise. they do not provide direct access to the hardware, but are simply context-less operations performed by the server.

XDGASync() blocks until all server rendering to the framebuffer completes. If Xlib or the 3 rendering functions above are used, **XDGASync**() must be called before the client directly accesses the framebuffer as the server rendering is asynchronous with the client and may have not completed. This is especially important if the **XDGAConcurrentAccess** flag is not set in the *XDGAMode*'s *flags* field since concurrent access by the server and client may result in a system lockup.

SEE ALSO

XFree86(1), XF86Config(5)

AUTHORS

XFree86-DGA version 2 was written by Mark Vojkovich. Version 1 was written by Jon Tombs, Harm Hanemaayer, Mark Vojkovich.

NAME

XF86MiscQueryExtension, XF86MiscQueryVersion, XF86MiscGetMouseSettings, XF86MiscSetMouseSettings, XF86MiscGetKbdSettings, XF86MiscSetKbdSettings – XFree86-Misc extension interface functions

SYNTAX

#include <X11/extensions/xf86misc.h>

Bool XF86MiscQueryExtension(
  Display **display*,
  int **event_base_return*,
  int **error_base_return*);

Bool XF86MiscQueryVersion(
  Display **display*,
  int **major_version_return*,
  int **minor_version_return*);

Status XF86MiscGetMouseSettings(
  Display **display*,
  XF86MiscMouseSettings **mseinfo*);

Status XF86MiscSetMouseSettings(
  Display **display*,
  XF86MiscMouseSettings **mseinfo*);

Status XF86MiscGetKbdSettings(
  Display **display*,
  XF86MiscKbdSettings **kbdinfo*);

Status XF86MiscSetKbdSettings(
  Display **display*,
  XF86MiscKbdSettings **kbdinfo*);

ARGUMENTS

display	Specifies the connection to the X server.
screen	Specifies which screen number the setting apply to.
event_base_return	Returns the base event number for the extension.
error_base_return	Returns the base error number for the extension.
major_version_return	Returns the major version number of the extension.
minor_version_return	Returns the minor version number of the extension.
mseinfo	Specifies a structure which contains the mouse parameters.
kbdinfo	Specifies a structure which contains the keyboard parameters.

STRUCTURES

Mouse:

```
typedef struct {
   char *device;                 /* returned path to device */
   int type;                     /* mouse protocol */
   int baudrate;                 /* 1200, 2400, 4800, or 9600 */
   int samplerate;               /* samples per second */
   int resolution;               /* resolution, count per inch */
   int buttons;                  /* number of buttons */
   Bool emulate3buttons;         /* Button1+Button3 -> Button2 ? */
   int emulate3timeout;          /* in milliseconds */
   Bool chordmiddle;             /* Button1+Button3 == Button2 ? */
   int flags;                    /* Device open flags */
```

```
} XF86MiscMouseSettings;
```

Keyboard:

```
typedef struct {
    int type;              /* of keyboard: 84-key, 101-key, Xqueue */
    int rate;              /* repeat rate */
    int delay;             /* delay until repeat starts */
    Bool servnumlock;      /* Server handles NumLock ? */
} XF86MiscKbdSettings;
```

DESCRIPTION

These functions provide an interface to the *XFree86-Misc* extension which allows various server settings to be queried and changed dynamically. Applications that use these functions must be linked with **-lXxf86misc**

POWER-SAVER FUNCTIONS

The **XF86MiscGetSaver** and **XF86MiscSetSaver** functions have been removed. This functionality is now provided by the DPMS extension.

MOUSE FUNCTIONS

Mouse parameters can be queried using the function **XF86MiscGetMouseSettings**. The structure pointed to by its second argument is filled in with the current mouse settings.

Not all fields are valid in all cases. For example, when the protocol indicates a bus mouse (i.e. the type field has value **MTYPE_BUSMOUSE** as defined in **xf86misc.h**), then the value in the **baudrate** field should be ignored as it does not apply to bus mice.

The **samplerate** field contains the resolution in lines per inch when using the Hitachi tablet protocol.

The device field of the structure points to dynamically allocated storage which should be freed by the caller.

Any of the fields of the structure can be altered and then passed to the **XF86MiscSetMouseSettings** function to change their value in the server, with the following restrictions:

1) The device can not be changed
2) The protocol can not be changed to or from Xqueue or OsMouse
3) The buttons field can not be changed
4) Invalid combinations of parameters are not allowed

The server will generate an error if any of the above is attempted, except the first – the contents of the device field are simply ignored.

A change of the protocol causes the device to be closed and reopened. Changes to the baud rate, sample rate, resolution or flags, when applicable to the selected protocol, also cause a reopen of the device. A reopen can be forced by using the MF_REOPEN flag, except in the case of the OsMouse and Xqueue protocols which ignore all attempts to reopen the device.

KEYBOARD FUNCTIONS

The **XF86MiscGetKbdSettings** function allows you to retrieve the current keyboard-related settings from the server.

Using the **XF86MiscSetKbdSettings** function, the keyboard autorepeat delay and rate can be set. Requests to change the **type** and **servnumlock** fields are ignored (except for checking for an invalid keyboard type). This is expected to change in a future release.

OTHER FUNCTIONS

Two functions, **XF86MiscQueryExtension** and **XF86MiscQueryVersion**, are provided which allow the client to query some information regarding the extension itself.

PREDEFINED VALUES

The header file **X11/extensions/xf86misc.h** contains definitions for

MTYPE_*	Mouse protocols
KTYPE_*	Keyboard types
MF_*	Mouse flags

SEE ALSO

xset(1), XF86Config(5)

AUTHORS

Joe Moss and David Dawes, The XFree86 Project, Inc.

NAME

XF86VidModeQueryExtension, XF86VidModeQueryVersion, XF86VidModeSetClientVersion, XF86VidModeGetModeLine, XF86VidModeGetAllModeLines, XF86VidModeDeleteModeLine, XF86VidModeModModeLine, XF86VidModeValidateModeLine, XF86VidModeSwitchMode, XF86VidModeSwitchToMode, XF86VidModeLockModeSwitch, XF86VidModeGetMonitor, XF86VidModeGetViewPort, XF86VidModeSetViewPort, XF86VidModeGetDotClocks, XF86VidModeGetGamma, XF86VidModeSetGamma, XF86VidModeGetGammaRamp, XF86VidModeSetGammaRamp, XF86VidModeGetGammaRampSize, XF86VidModeGetPermissions – XFree86-VidMode extension interface functions

SYNTAX

#include <X11/extensions/xf86vmode.h>

Bool XF86VidModeQueryExtension(
Display **display*,
int **event_base_return*,
int **error_base_return*);

Bool XF86VidModeQueryVersion(
Display **display*,
int **major_version_return*,
int **minor_version_return*);

Bool XF86VidModeSetClientVersion(
Display **display*);

Bool XF86VidModeGetModeLine(
Display **display*,
int *screen*,
int **dotclock_return*,
XF86VidModeModeLine **modeline*);

Bool XF86VidModeGetAllModeLines(
Display **display*,
int *screen*,
int **modecount_return*,
XF86VidModeModeInfo ****modesinfo*);

Bool XF86VidModeDeleteModeLine(
Display **display*,
int *screen*,
XF86VidModeModeInfo **modeline*);

Bool XF86VidModeModModeLine(
Display **display*,
int *screen*,
XF86VidModeModeLine **modeline*);

Status XF86VidModeValidateModeLine(
Display **display*,
int *screen*,
XF86VidModeModeLine **modeline*);

Bool XF86VidModeSwitchMode(
Display **display*,
int *screen*,
int *zoom*);

Bool XF86VidModeSwitchToMode(
Display **display*,
int *screen*,
XF86VidModeModeInfo **modeline*);

Bool XF86VidModeLockModeSwitch(
  Display **display*,
  int *screen*,
  int *lock*);

Bool XF86VidModeGetMonitor(
  Display **display*,
  int *screen*,
  XF86VidModeMonitor **monitor*);

Bool XF86VidModeGetViewPort(
  Display **display*,
  int *screen*,
  int **x_return*,
  int **y_return*);

Bool XF86VidModeSetViewPort(
  Display **display*,
  int *screen*,
  int *x*,
  int *y*);

XF86VidModeGetDotClocks(
  Display **display*,
  int *screen*,
  int **flags return*,
  int **number of clocks return*,
  int **max dot clock return*,
  int ***clocks return*);

XF86VidModeGetGamma(
  Display **display*,
  int *screen*,
  XF86VidModeGamma **Gamma*);

XF86VidModeSetGamma(
  Display **display*,
  int *screen*,
  XF86VidModeGamma **Gamma*);

XF86VidModeGetGammaRamp(
  Display **display*,
  int *screen*,
  int *size*,
  unsigned short **red array*,
  unsigned short **green array*,
  unsigned short **blue array*);

XF86VidModeSetGammaRamp(
  Display **display*,
  int *screen*,
  int *size*,
  unsigned short **red array*,
  unsigned short **green array*,
  unsigned short **blue array*);

XF86VidModeGetGammaRampSize(
  Display **display*,
  int *screen*,
  int **size*);

ARGUMENTS

display	Specifies the connection to the X server.
screen	Specifies which screen number the setting apply to.
event_base_return	Returns the base event number for the extension.
error_base_return	Returns the base error number for the extension.
major_version_return	Returns the major version number of the extension.
minor_version_return	Returns the minor version number of the extension.
dotclock_return	Returns the clock for the mode line.
modecount_return	Returns the number of video modes available in the server.
zoom	If greater than zero, indicates that the server should switch to the next mode, otherwise switch to the previous mode.
lock	Indicates that mode switching should be locked, if non-zero.
modeline	Specifies or returns the timing values for a video mode.
modesinfo	Returns the timing values and dotclocks for all of the available video modes.
monitor	Returns information about the monitor.
x	Specifies the desired X location for the viewport.
x_return	Returns the current X location of the viewport.
y	Specifies the desired Y location for the viewport.
y_return	Returns the current Y location of the viewport.

STRUCTURES

Video Mode Settings:

```
typedef struct {
   unsigned short     hdisplay;      /* Number of display pixels horizontally */
   unsigned short     hsyncstart;    /* Horizontal sync start */
   unsigned short     hsyncend;      /* Horizontal sync end */
   unsigned short     htotal;        /* Total horizontal pixels */
   unsigned short     vdisplay;      /* Number of display pixels vertically */
   unsigned short     vsyncstart;    /* Vertical sync start */
   unsigned short     vsyncend;      /* Vertical sync start */
   unsigned short     vtotal;        /* Total vertical pixels */
   unsigned int       flags;         /* Mode flags */
   int                privsize;      /* Size of private */
   INT32              *private;      /* Server privates */
} XF86VidModeModeLine;

typedef struct {
   unsigned int       dotclock;      /* Pixel clock */
   unsigned short     hdisplay;      /* Number of display pixels horizontally */
   unsigned short     hsyncstart;    /* Horizontal sync start */
   unsigned short     hsyncend;      /* Horizontal sync end */
   unsigned short     htotal;        /* Total horizontal pixels */
   unsigned short     vdisplay;      /* Number of display pixels vertically */
   unsigned short     vsyncstart;    /* Vertical sync start */
   unsigned short     vsyncend;      /* Vertical sync start */
   unsigned short     vtotal;        /* Total vertical pixels */
   unsigned int       flags;         /* Mode flags */
   int                privsize;      /* Size of private */
```

```
    INT32                   *private;       /* Server privates */
} XF86VidModeModeInfo;
```

Monitor information:

```
typedef struct {
    char*                   vendor;         /* Name of manufacturer */
    char*                   model;          /* Model name */
    float                   EMPTY;          /* unused, for backward compatibility */
    unsigned char           nhsync;         /* Number of horiz sync ranges */
    XF86VidModeSyncRange*   hsync;          /* Horizontal sync ranges */
    unsigned char           nvsync;         /* Number of vert sync ranges */
    XF86VidModeSyncRange*   vsync;          /* Vertical sync ranges */
} XF86VidModeMonitor;

typedef struct {
    float                   hi;             /* Top of range */
    float                   lo;             /* Bottom of range */
} XF86VidModeSyncRange;

typedef struct {
    int type;                               /* of event */
    unsigned long serial;   /* # of last request processed by server */
    Bool send_event;                        /* true if this came from a SendEvent req */
    Display *display;                       /* Display the event was read from */
    Window root;                            /* root window of event screen */
    int state;                              /* What happened */
    int kind;                               /* What happened */
    Bool forced;                            /* extents of new region */
    Time time;                              /* event timestamp */
} XF86VidModeNotifyEvent;

typedef struct {
    float red;                              /* Red Gamma value */
    float green;                            /* Green Gamma value */
    float blue;                             /* Blue Gamma value */
} XF86VidModeGamma;
```

DESCRIPTION

These functions provide an interface to the server extension *XFree86-VidModeExtension* which allows the video modes to be queried and adjusted dynamically and mode switching to be controlled. Applications that use these functions must be linked with **-lXxf86vm**

MODELINE FUNCTIONS

The **XF86VidModeGetModeLine** function is used to query the settings for the currently selected video mode. The calling program should pass a pointer to a **XF86VidModeModeLine** structure that it has already allocated. The function fills in the fields of the structure.

If there are any server private values (currently only applicable to the S3 server) the function will allocate storage for them. Therefore, if the **privsize** field is non-zero, the calling program should call **Xfree(private)** to free the storage.

XF86VidModeGetAllModeLines returns the settings for all video modes. The calling program supplies the address of a pointer which will be set by the function to point to an array of **XF86VidModeModeInfo** structures. The memory occupied by the array is dynamically allocated by the **XF86VidModeGetAllModeLines** function and should be freed by the caller. The first element of the array corresponds to the current video mode.

The **XF86VidModeModModeLine** function can be used to change the settings of the current video mode provided the requested settings are valid (e.g. they don't exceed the capabilities of the monitor).

Modes can be deleted with the **XF86VidModeDeleteModeLine** function. The specified mode must match an existing mode. To be considered a match, all of the fields of the given **XF86VidModeModeInfo** structure must match, except the **privsize** and **private** fields. If the mode to be deleted is the current mode, a mode switch to the next mode will occur first. The last remaining mode can not be deleted.

The validity of a mode can be checked with the **XF86VidModeValidateModeLine** function. If the specified mode can be used by the server (i.e. meets all the constraints placed upon a mode by the combination of the server, card, and monitor) the function returns **MODE_OK**, otherwise it returns a value indicating the reason why the mode is invalid (as defined in *xf86.h*)

MODE SWITCH FUNCTIONS

When the function **XF86VidModeSwitchMode** is called, the server will change the video mode to next (or previous) video mode. The **XF86VidModeSwitchToMode** function can be used to switch directly to the specified mode. Matching is as specified in the description of the **XF86VidModeAddModeLine** function above. The **XF86VidModeLockModeSwitch** function can be used to allow or disallow mode switching whether the request to switch modes comes from a call to the **XF86VidModeSwitchMode** or **XF86VidModeSwitchToMode** functions or from one of the mode switch key sequences.

Note: Because of the asynchronous nature of the X protocol, a call to **XFlush** is needed if the application wants to see the mode change immediately. To be informed of the execution status of the request, a custom error handler should be installed using **XSetErrorHandler** before calling the mode switching function.

MONITOR FUNCTIONS

Information known to the server about the monitor is returned by the **XF86VidModeGetMonitor** function. The **hsync** and **vsync** fields each point to an array of **XF86VidModeSyncRange** structures. The arrays contain **nhsync** and **nvsync** elements, respectively. The **hi** and **low** values will be equal if a discreate value was given in the **XF86Config** file.

The **vendor**, **model**, **hsync**, and **vsync** fields point to dynamically allocated storage that should be freed by the caller.

VIEWPORT FUNCTIONS

The **XF86VidModeGetViewPort** and **XF86VidModeSetViewPort** functions can be used to, respectively, query and change the location of the upper left corner of the viewport into the virtual screen.

OTHER FUNCTIONS

The **XF86VidModeQueryVersion** function can be used to determine the version of the extension built into the server.

The function **XF86VidModeQueryExtension** returns the lowest numbered error and event values assigned to the extension.

BUGS

The XF86VidModeSetClientVersion, XF86VidModeGetDotClocks, XF86VidModeGetGamma, XF86VidModeSetGamma, XF86VidModeSetGammaRamp, XF86VidModeGetGammaRamp, XF86VidModeGetGammaRampSize, and XF86VidModeGetPermissions functions need to be documented. In the meantime, check the source code for information about how to use them.

SEE ALSO

XFree86(1), XF86Config(5), XFlush(3), XSetErrorHandler(3), xvidtune(1)

AUTHORS

Kaleb Keithley, Jon Tombs, David Dawes, and Joe Moss

NAME

X − a portable, network-transparent window system

SYNOPSIS

The X Window System is a network transparent window system which runs on a wide range of computing and graphics machines. It should be relatively straightforward to build the X Consortium 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 X Consortium, Inc.

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 X Consortium distribution include: a terminal emulator, *xterm*; a window manager, *twm*; a display manager, *xdm*; a console redirect program, *xconsole*; a mail interface, *xmh*; a bitmap editor, *bitmap*; resource listing/manipulation tools, *appres*, *editres*; access control programs, *xauth*, *xhost*, and *iceauth*; user preference setting programs, *xrdb*, *xcmsdb*, *xset*, *xsetroot*, *xstdcmap*, and *xmodmap*; clocks, *xclock* and *oclock*; a font displayer, (*xfd*; utilities for listing information about fonts, windows, and displays, *xlsfonts*, *xwininfo*, *xlsclients*, *xdpyinfo*, *xlsatoms*, and *xprop*; screen image manipulation utilities, *xwd*, *xwud*, and *xmag*; a performance measurement utility, *x11perf*; a font compiler, *bdftopcf*; a font server and related utilities, *xfs*, *fsinfo*, *fslsfonts*, *fstobdf*; an X Image Extension exerciser, *xieperf*; a display server and related utilities, *Xserver*, *rgb*, *mkfontdir*; remote execution utilities, *rstart* and *xon*; a clipboard manager, *xclipboard*; keyboard description compiler and related utilities, *xkbcomp*, *xkbprint*, *xkbbell*, *xkbevd*, *xkbvleds*, and *xkbwatch*; a utility to terminate clients, *xkill*; an optimized X protocol proxy, *lbxproxy*; a firewall security proxy, *xfwp*; a proxy manager to control them, *proxymngr*; a utility to find proxies, *xfindproxy*; Netscape Navigator Plug-ins, *libxrx.so* and *libxrxnest.so*; an RX MIME-type helper program, *xrx*; and a utility to cause part or all of the screen to be redrawn, *xrefresh*.

Many other utilities, window managers, games, toolkits, etc. are included as user-contributed software in the X Consortium 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 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 perspective, 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, 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, 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 will 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.

TCPIP
: 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: *x.org:0*, *expo:0*, *198.112.45.11:0*, *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 6 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.
MIT-KERBEROS-5	Kerberos Version 5 user-to-user.

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, 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 is placed in roughly the center of the screen and a load average monitor, mailbox, and clock are 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 X Consortium 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 X Consortium 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.

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.

FONT SERVER NAMES

One of the following forms can be used to name a font server that accepts TCP connections:

tcp/*hostname*:*port*
tcp/*hostname*:*port*/*cataloguelist*

The *hostname* specifies the name (or decimal numeric address) of the machine on which the font server is running. The *port* is the decimal TCP port on which the font server is listening for connections. The *cataloguelist* specifies a list of catalogue names, with '+' as a separator.

Examples: *tcp/x.org:7100*, *tcp/198.112.45.11:7100/all*.

One of the following forms can be used to name a font server that accepts DECnet connections:

decnet/*nodename*::font$*objname*
decnet/*nodename*::font$*objname*/*cataloguelist*

The *nodename* specifies the name (or decimal numeric address) of the machine on which the font server is running. The *objname* is a normal, case-insensitive DECnet object name. The *cataloguelist* specifies a list of catalogue names, with '+' as a separator.

Examples: *DECnet/SRVNOD::FONT$DEFAULT*, *decnet/44.70::font$special/symbols*.

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 *__projectroot__/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 (which names a single directory and should end with a '/' on POSIX systems), plus directories in a standard place (usually under /usr/X11R6/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/X11R6/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.

ENVIRONMENT

DISPLAY
This is the only mandatory environment variable. It must point to an X server. See section "Display Names" above.

XAUTHORITY
This must point to a file that contains authorization data. The default is *$HOME/.Xauthority*. See **Xsecurity**(7), **xauth**(1), **xdm**(1), **Xau**(3).

ICEAUTHORITY
This must point to a file that contains authorization data. The default is *$HOME/.ICEauthority*.

LC_ALL, **LC_CTYPE**, **LANG**
The first non-empty value among these three determines the current locale's facet for character handling, and in particular the default text encoding. See **locale**(7), **setlocale**(3), **locale**(1).

XMODIFIERS
This variable can be set to contain additional information important for the current locale setting. Typically set to *@im=<input-method>* to enable a particular input method. See **XSetLocaleModifiers**(3).

XLOCALEDIR
This must point to a directory containing the locale.alias file and Compose and XLC_LOCALE file hierarchies for all locales. The default value is *__projectroot__/lib/X11/locale*.

XENVIRONMENT
This must point to a file containing X resources. The default is *$HOME/.Xdefaults-<hostname>*. Unlike *__projectroot__/lib/X11/Xresources*, it is consulted each time an X application starts.

XFILESEARCHPATH
This must contain a colon separated list of path templates, where libXt will search for resource files. The default value consists of

```
/usr/X11R6/lib/X11/%L/%T/%N%C%S:\
/usr/X11R6/lib/X11/%l/%T/%N%C%S:\
/usr/X11R6/lib/X11/%T/%N%C%S:\
/usr/X11R6/lib/X11/%L/%T/%N%S:\
/usr/X11R6/lib/X11/%l/%T/%N%S:\
/usr/X11R6/lib/X11/%T/%N%S
```

A path template is transformed to a pathname by substituting:

```
%N => name (basename) being searched for
%T => type (dirname) being searched for
%S => suffix being searched for
%C => value of the resource "customization"
     (class "Customization")
%L => the locale name
%l => the locale's language (part before '_')
%t => the locale's territory (part after '_' but before '.')
%c => the locale's encoding (part after '.')
```

XUSERFILESEARCHPATH
This must contain a colon separated list of path templates, where libXt will search for user dependent resource files. The default value is:

```
$XAPPLRESDIR/%L/%N%C:\
$XAPPLRESDIR/%l/%N%C:\
```

```
$XAPPLRESDIR/%N%C:\
$HOME/%N%C:\
$XAPPLRESDIR/%L/%N:\
$XAPPLRESDIR/%l/%N:\
$XAPPLRESDIR/%N:\
$HOME/%N
```

$XAPPLRESDIR defaults to *$HOME*, see below.

A path template is transformed to a pathname by substituting:

```
%N => name (basename) being searched for
%T => type (dirname) being searched for
%S => suffix being searched for
%C => value of the resource "customization"
     (class "Customization")
%L => the locale name
%l => the locale's language (part before '_')
%t => the locale's territory (part after '_' but before '.')
%c => the locale's encoding (part after '.')
```

XAPPLRESDIR
This must point to a base directory where the user stores his application dependent resource files. The default value is *$HOME*. Only used if XUSERFILESEARCHPATH is not set.

XKEYSYMDB
This must point to a file containing nonstandard keysym definitions. The default value is *__projectroot__/lib/X11/XKeysymDB*.

XCMSDB
This must point to a color name database file. The default value is *__projectroot__/lib/X11/Xcms.txt*.

XFT_CONFIG
This must point to a configuration file for the Xft library. The default value is *__projectroot__/lib/X11/XftConfig*.

RESOURCE_NAME
This serves as main identifier for resources belonging to the program being executed. It defaults to the basename of pathname of the program.

SESSION_MANAGER
Denotes the session manager the application should connect. See **xsm**(1), **rstart**(1).

XF86BIGFONT_DISABLE
Setting this variable to a non-empty value disables the XFree86-Bigfont extension. This extension is a mechanism to reduce the memory consumption of big fonts by use of shared memory.

XKB_FORCE
XKB_DISABLE
XKB_DEBUG
_XKB_CHARSET
_XKB_LOCALE_CHARSETS
_XKB_OPTIONS_ENABLE
_XKB_LATIN1_LOOKUP
_XKB_CONSUME_LOOKUP_MODS
_XKB_CONSUME_SHIFT_AND_LOCK
_XKB_IGNORE_NEW_KEYBOARDS
_XKB_CONTROL_FALLBACK

_XKB_COMP_LED _XKB_COMP_FAIL_BEEP

These variables influence the X Keyboard Extension.

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 *__projectroot__/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

XProjectTeam(7), **XStandards**(7), **Xsecurity**(7), **appres**(1), **bdftopcf**(1), **bitmap**(1), **editres**(1), **fsinfo**(1), **fslsfonts**(1), **fstobdf**(1), **iceauth**(1), **imake**(1), **lbxproxy**(1), **makedepend**(1), **mkfontdir**(1),

oclock(1), **proxymngr**(1), **rgb**(1), **resize**(1), **rstart**(1), **smproxy**(1), **twm**(1), **x11perf**(1), **x11perfcomp**(1), **xauth**(1), **xclipboard**(1), **xclock**(1), **xcmsdb**(1), **xconsole**(1), **xdm**(1), **xdpyinfo**(1), **xfd**(1), **xfindproxy**(1), **xfs**(1), **xfwp**(1), **xhost**(1), **xieperf**(1), **xinit**(1), **xkbbell**(1), **xkbcomp**(1), **xkbevd**(1), **xkbprint**(1), **xkbvleds**(1), **xkbwatch**(1), **xkill**(1), **xlogo**(1), **xlsatoms**(1), **xlsclients**(1), **xlsfonts**(1), **xmag**(1), **xmh**(1), **xmodmap**(1), **xon**(1), **xprop**(1), **xrdb**(1), **xrefresh**(1), **xrx**(1), **xset**(1), **xsetroot**(1), **xsm**(1), **xstdcmap**(1), **xterm**(1), **xwd**(1), **xwininfo**(1), **xwud**(1). **Xserver**(1), **Xdec**(1), **XmacII**(1), **Xsun**(1), **Xnest**(1), **Xvfb**(1), **XFree86**(1), **XDarwin**(1), **kbd_mode**(1), *Xlib − C Language X Interface*, and *X Toolkit Intrinsics − C Language Interface*

TRADEMARKS

X Window System is a trademark of X Consortium, Inc.

AUTHORS

A cast of thousands, literally. The Release 6.3 distribution is brought to you by X Consortium, Inc. The names of all people who made it a reality will be found in the individual documents and source files. The staff members at the X Consortium responsible for this release are: Donna Converse (emeritus), Stephen Gildea (emeritus), Kaleb Keithley, Matt Landau (emeritus), Ralph Mor (emeritus), Janet O'Halloran, Bob Scheifler, Ralph Swick, Dave Wiggins (emeritus), and Reed Augliere.

The X Window System standard was originally developed at the Laboratory for Computer Science at the Massachusetts Institute of Technology, and all rights thereto were assigned to the X Consortium on January 1, 1994. X Consortium, Inc. closed its doors on December 31, 1996. All rights to the X Window System have been assigned to the Open Software Foundation.

NAME

XStandards − X Consortium Standards and X Project Team Specifications

SYNOPSIS

The major goal of the X Consortium was to promote cooperation within the computer industry in the creation of standard software interfaces at all layers in the X Window System environment. The X Consortium produced standards - documents which defined network protocols, programming interfaces, and other aspects of the X environment. These standards continue to exist in The Open Group's X Project Team releases. The X Project Team produces specifications. Like X Consortium Standards, these are documents which define network protocols, programming interfaces, and other aspects of the X environment. Under the aegis of The Open Group, X Consortium standards, X Project Team specifications, and other specifications are the basis for portions of The Open Group's various CAE specifications.

The status of various standards, specifications, and the software in the X11R6.4 distribution, is explained below.

STANDARDS

The following documents are X Consortium standards:

X Window System Protocol
X Version 11, Release 6.4
Robert W. Scheifler

Xlib − C Language X Interface
X Version 11, Release 6.4
James Gettys, Robert W. Scheifler, Ron Newman

X Toolkit Intrinsics − C Language Interface
X Version 11, Release 6.4
Joel McCormack, Paul Asente, Ralph R. Swick, Donna Converse

Bitmap Distribution Format
Version 2.1
X Version 11, Release 6.4

Inter-Client Communication Conventions Manual
Version 2.0
X Version 11, Release 6.4
David Rosenthal, Stuart W. Marks

Compound Text Encoding
Version 1.1
X Version 11, Release 6.4
Robert W. Scheifler

X Logical Font Description Conventions
Version 1.5
X Version 11, Release 6.4
Jim Flowers, Stephen Gildea

X Display Manager Control Protocol
Version 1.0
X Version 11, Release 6.4
Keith Packard

X11 Nonrectangular Window Shape Extension

Version 1.0
X Version 11, Release 6.4
Keith Packard

X11 Input Extension Protocol Specification
Version 1.0
X Version 11, Release 6.4
George Sachs, Mark Patrick

X11 Input Extension Library Specification
X Version 11, Release 6.4
Mark Patrick, George Sachs

The X Font Service Protocol
Version 2.0
X Version 11, Release 6.4
Jim Fulton

PEX Protocol Specification
Version 5.1
Cheryl Huntington (architect), Paula Womack (editor)

PEXlib Specification and C Language Binding
Version 5.1
Jeff Stevenson

Inter-Client Exchange (ICE) Protocol
Version 1.0
X Version 11, Release 6.4
Robert Scheifler, Jordan Brown

Inter-Client Exchange (ICE) Library
Version 1.0
X Version 11, Release 6.4
Ralph Mor

X Session Management Protocol
Version 1.0
X Version 11, Release 6.4
Mike Wexler

X Session Management Library
Version 1.0
X Version 11, Release 6.4
Ralph Mor

The Input Method Protocol
Version 1.0
X Version 11, Release 6.4
Masahiko Narita, Hideki Hiura

X Synchronization Extension
Version 3.0
X Version 11, Release 6.4

Tim Glauert, Dave Carver, Jim Gettys, David P. Wiggins

X Image Extension, Protocol Reference Manual
Version 5.0
X Version 11, Release 6.4
Bob Shelley

XTEST Extension
Version 2.2
Kieron Drake

Big Requests Extension
Version 2.0
X Version 11, Release 6.4
Bob Scheifler

XC-MISC Extension
Version 1.1
X Version 11, Release 6.4
Bob Scheifler, Dave Wiggins

Double Buffer Extension
Version 1.0
Ian Elliott, David P. Wiggins

Record Extension Protocol
Version 1.13
Martha Zimet, Stephen Gildea

Record Extension Library
Version 1.13
Martha Zimet, Stephen Gildea

X Keyboard Extension Protocol
X Version 11, Release 6.4
Erik Fortune

X Keyboard Extension Library
X Version 11, Release 6.4
Amber J. Benson, Gary Aitken, Erik Fortune, Donna Converse,
George Sachs, and Will Walker

X Print Extension Protocol
X Version 11, Release 6.4

X Print Extension Library
X Version 11, Release 6.4

X Application Group Extension Protocol and Library
Version 1.0
X Version 11, Release 6.4
Kaleb Keithley

X Security Extension Protocol and Library

Version 4.0
X Version 11, Release 6.4
Dave Wiggins

X Proxy Manager Protocol
X Version 11, Release 6.4
Ralph Swick

LBX Extension Protocol and Library
X Version 11, Release 6.4
Keith Packard, Dave Lemke, Donna Converse, Ralph Mor, Ray Tice

Remote Execution MIME Type
Version 1.0
X Version 11, Release 6.4
Arnaud Le Hors

SPECIFICATIONS

The following documents are X Project Team specifications:

Colormap Utilization Policy and Extension
Version 1.0
Kaleb Keithley

Extended Visual Information Extension
Version 1.0
Peter Daifuku

X Display Power Management (DPMS) Extension Protocol and Library
Version 1.0
Rob Lembree

DRAFT STANDARDS

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

X Image Extension Library
Public Review Draft
Gary Rogers

PEX Protocol Specification
Version 5.2
Jeff Stevenson (architect), Jane Sczechowski (editor)

PEXlib Specification and C Language Binding
Version 5.2
Karl Schultz

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 Input Extension standard.

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

The following include files are part of the PEXlib standard.

<X11/PEX5/PEX.h>
<X11/PEX5/PEXlib.h>
<X11/PEX5/PEXlibint.h>
<X11/PEX5/PEXproto.h>
<X11/PEX5/PEXprotost.h>

The following include files are part of the ICElib standard.

<X11/ICE/ICE.h>
<X11/ICE/ICEconn.h>
<X11/ICE/ICElib.h>
<X11/ICE/ICEmsg.h>
<X11/ICE/ICEproto.h>
<X11/ICE/ICEutil.h>

The following include files are part of the SMlib standard.

<X11/SM/SM.h>
<X11/SM/SMlib.h>
<X11/SM/SMproto.h>

The following include file is part of the Synchronization standard.

<X11/extensions/sync.h>

The following include files are part of the XIElib draft standard.

<X11/extensions/XIE.h>
<X11/extensions/XIElib.h>
<X11/extensions/XIEproto.h>
<X11/extensions/XIEprotost.h>

The following include file is part of the XTEST standard.

<X11/extensions/XTest.h>

The following include file is part of the Double Buffer Extension standard.

<X11/extensions/Xdbe.h>

The following include file is part of the Record Library standard.

<X11/extensions/record.h>

The following include files are part of the X Keyboard Extension Library standard.

<X11/XKBlib.h>
<X11/extensions/XKB.h>
<X11/extensions/XKBproto.h>
<X11/extensions/XKBstr.h>
<X11/extensions/XKBgeom.h>

The following include files are part of the X Print Extension Library standard.

<X11/extensions/Print.h>
<X11/extensions/Printstr.h>

The following include files are part of the X Application Group Extension Library standard.

<X11/extensions/Xag.h>
<X11/extensions/Xagstr.h>

The following include files are part of the X Security Extension Library standard.

<X11/extensions/security.h>
<X11/extensions/securstr.h>

The following include files are part of the LBX Extension library standard.

<X11/extensions/XLbx.h>
<X11/extensions/lbxbuf.h>
<X11/extensions/lbxbufstr.h>
<X11/extensions/lbxdeltastr.h>
<X11/extensions/lbximage.h>
<X11/extensions/lbxopts.h>
<X11/extensions/lbxstr.h>
<X11/extensions/lbxzlib.h>

The following include files are part of the Colormap Utilization Policy and Extention specification.

<X11/extensions/Xcup.h>
<X11/extensions/Xcupstr.h>

The following include files are part of the Extended Visual Information specification.

<X11/extensions/XEVI.h>
<X11/extensions/XEVIstr.h>

The following include files are part of the X Display Management Signaling Extension specification.

<X11/extensions/dpms.h>
<X11/extensions/dpmsstr.h>

NON STANDARDS

The X11R6.4 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.

The only X Consortium standards are the ones listed above. No other documents, include files, or software in X11R6.4 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 Athena Widget Set; the Xmu library; the Xau library; the RGB database; the X Locale database; the fonts distributed with X11R6.4; the applications distributed with X11R6.4; the include files <X11/XWDFile.h>, <X11/Xfuncproto.h>, <X11/Xfuncs.h>, <X11/Xosdefs.h>, <X11/Xos.h>, <X11/Xos_r.h>, <X11/Xwinsock.h>, and <X11/Xthreads.h>; the bitmap files in <X11/bitmaps>.

The Multi-Buffering extension was a draft standard of the X Consortium but has been superseded by DBE as a standard.

X REGISTRY

The X Consortium maintained a registry of certain X-related items, to aid in avoiding conflicts and to aid in sharing of such items.

The registry is published as part of the X Consortium software release. The latest version may also be available by sending a message to xstuff@x.org. The message can have a subject line and no body, or a single-line body and no subject, in either case the line looking like:

    send docs registry

The X Registry and the names in it are not X Consortium standards.

NAME

XConsortium − X Consortium information

SYNOPSIS

Release 6.3 of X Version 11 was brought to you by X Consortium, Inc.

DESCRIPTION

The X Consortium was an independent, not-for-profit Delaware membership corporation. It was formed in 1993 as the successor to the MIT X Consortium. The purpose of the X Consortium was to foster the development, evolution, and maintenance of the X Window System, a comprehensive set of vendor-neutral, system-architecture neutral, network-transparent windowing and user interface standards.

The X Window System was created in the mid-1980s at the Massachusetts Institute of Technology. In 1988, MIT formed a member-funded consortium to provide the technical and administrative leadership necessary to support further development of the X Window System. In 1992, MIT and the membership decided it was in their best interests to move the consortium out of MIT and create an independent, stand-alone organization. All rights to the X Window System were assigned by MIT to X Consortium, Inc. on January 1, 1994. X Consortium, Inc. closed its doors on December 31, 1996. All rights to the X Window System have been assigned to the Open Software Foundation.

The X Consortium was financially self-supporting through membership fees. There are no license fees associated with the use of X Window System standards and code developed by the X Consortium. Membership in the X Consortium was open to any organization willing to execute a membership agreement.

The X Consortium was a highly participative body. Members were encouraged to actively cooperate with the staff and other members in the design and review of proposed specifications, and in the design, coding and testing of sample implementations of those specifications.

The X Consortium accomplished most of its work using electronic mail over the Internet, with individual mailing lists for working groups. Internet electronic mail connectivity was viewed as a requirement for useful participation in X Consortium activities. Meetings were held as necessary, often in conjunction with industry conferences and trade shows.

STAFF

President:
Bob Scheifler

Office Manager:
Janet O'Halloran

Director of Marketing:
Paul Lavallee

Director of Engineering:
Jim Fournier

Manager, X Window System:
Matt Landau, emeritus

Technical Director, X Window System:
Ralph Swick

Technical Staff, X Window System:

Donna Converse, emeritus
Stephen Gildea, emeritus
Kaleb Keithley
Arnaud Le Hors
Ralph Mor, emeritus

Ray Tice
Dave Wiggins, emeritus

Managers, CDE Development:
Giora Guth
Peter Bohnert, emeritus

Manager, CDE Quality Engineering:
David Brooks

CDE Architects:
Kevin Samborn
Daniel Dardailler, emeritus

Technical Staff, CDE Development:

Art Barstow
Pascale Dardailler
David Kaelbling
Mitch Greess
Robert Seacord

Technical Staff, CDE Quality Engineering:

Chris Burleson
Tom Cavin
Sami Mohammed
Mark Schuldenfrei

Manager, Systems Administration:
Kevin Ethier

Technical Staff, Systems Administration:
Mike Donati
Amy Rich, emeritus
Anne Salemme

BOARD OF DIRECTORS

The X Consortium's activities and affairs were managed under the direction and oversight of a Board of Directors, elected annually by the Members. The Board was responsible for reviewing the achievements of the Consortium, approving planned work, appointing a President and other officers of the Consortium, and setting membership dues. The last Directors were:

Robert W. Scheifler, President, X Consortium
Dr. Forest Baskett, Senior VP of R&D, Silicon Graphics Computer Systems
Harold D. Blair, Apogee International Corp.
Roger S. Gourd, Gourd & Associates
Dr. Robin Hillyard, Chairman and Chief Technical Officer, Novasoft Systems
Don McGovern, General Operations Manager and Executive Dir., Hewlett Packard
Peter J. Shaw, Senior VP, NetManage
Michael Tobias, President, Tech-Source, Inc.

ADDRESS

To reach the X Consortium public Wide World Web server, use the URL: http://www.x.org/

To reach the X Consortium public ftp machine, use anonymous ftp to: ftp.x.org

FULL MEMBERS

Adobe Systems Inc.
Cray Research, Inc.
Digital Equipment Corp.
Fujitsu Limited
Hewlett-Packard Company
Hitachi Ltd.
IBM Corporation
Megatek Corp.
Motorola, Inc.
NEC Corporation
Novell, Inc.
Oki Electric Industry Co., Ltd.
OMRON Corporation
SCO, Inc.
Siemens Nixdorf Informationssysteme AG
Silicon Graphics, Inc.
Sony Corporation
Sun Microsystems, Inc.
Tektronix, Inc.

ASSOCIATE MEMBERS

Boundless Technologies
Hummingbird Communications Ltd.
Insignia Solutions, Ltd.
Mercury Interactive Corp.
NetManage, Inc.
Network Computing Devices
VisiCom Laboratories, Inc.
Walker Richer & Quinn, Inc.

END USERS

Hughes Aircraft Company

AFFILIATE MEMBERS

ASTEC, Inc.
BARCO Chromatics, Inc.
CenterLine Software, Inc.
CliniComp, Intl.
Component Integration Laboratories, Inc.
Draper Laboratory.
Electronic Book Technologies, Inc.
Gallium Software, Inc.
Georgia Institute of Technology
Human Designed Systems, Inc.
INRIA − Institut National de Recherche en Informatique et en Automatique
Integrated Computer Solutions, Inc.
Investment Management Services, Inc.

Jupiter Systems
KL Group Inc.
Massachusetts Institute of Technology
Metheus Corporation
Metro Link, Inc.
Object Management Group, Inc.
Open Software Foundation
Performance Awareness Corp.
Peritek Corp.
Petrotechnical Open Software Corp.
Point Technologies, Inc.
Shiman Associates, Inc.
Smithsonian Astrophysical Observatory.
Software Development Corp.
SOUM Corporation
Spectragraphics Corp.
Tech-Source, Inc.
TriTeal Corp.
White Pine Software, Inc.
World Wide Web Consortium.
The XFree86 Project, Inc.
X Inside, Inc.

NAME

Xsecurity − X display access control

SYNOPSIS

X provides mechanism for implementing many access control systems. The sample implementation includes five 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.
MIT-KERBEROS-5	Kerberos Version 5 user-to-user.

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

Sites who compile with DES support can use 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/IPv4 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 server’s 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@ ruth@mit.edu
adds "keith" from the NIS domain of the local machine, and "ruth" in the "mit.edu" NIS domain. For keith or ruth 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@our.domain.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. To access the display from a remote host, you may have to do a *keylogin* on the remote host first.

MIT-KERBEROS-5

Kerberos is a network-based authentication scheme developed by MIT for Project Athena. It allows mutually suspicious principals to authenticate each other as long as each trusts a third party, Kerberos. Each principal has a secret key known only to it and Kerberos. Principals includes servers, such as an FTP server or X server, and human users, whose key is their password. Users gain access to services by getting Kerberos tickets for those services from a Kerberos server. Since the X server has no place to store a secret key, it shares keys with the user who logs in. X authentication thus uses the user-to-user scheme of Kerberos version 5.

When you log in via *xdm*, *xdm* will use your password to obtain the initial Kerberos tickets. *xdm* stores the tickets in a credentials cache file and sets the environment variable *KRB5CCNAME* to point to the file. The credentials cache is destroyed when the session ends to reduce the chance of the tickets being stolen before they expire.

Since Kerberos is a user-based authorization protocol, like the SUN-DES-1 protocol, the owner of a display can enable and disable specific users, or Kerberos principals. The *xhost* client is used to enable or disable authorization. For example,
xhost krb5:judy krb5:gildea@x.org
adds "judy" from the Kerberos realm of the local machine, and "gildea" from the "x.org" realm.

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 and MIT-KERBEROS-5 use 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. This information is used to encrypt the client's authority information when it is sent to the 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@our.domain.edu"). Putting the correct principal name in the *.Xauthority* file causes Xlib to generate the appropriate authorization information using the secure RPC library.

MIT-KERBEROS-5
: Kerberos reads tickets from the cache pointed to by the *KRB5CCNAME* environment variable, so does not use any data from the *.Xauthority* file. An entry with no data must still exist to tell clients that MIT-KERBEROS-5 is available.

: Unlike the *.Xauthority* file for clients, the authority file passed by xdm to a local X server (with ‘‘**−auth** *filename*’’, see xdm(1)) does contain the name of the credentials cache, since the X server will not have the *KRB5CCNAME* environment variable set. The data of the MIT-KERBEROS-5 entry is the credentials cache name and has the form ‘‘UU:FILE:*filename*’’, where *filename* is the name of the credentials cache file created by xdm. Note again that this form is *not* used by clients.

FILES

.Xauthority

SEE ALSO

X(7), xdm(1), xauth(1), xhost(1), xinit(1), Xserver(1)

NAME

appres − list X 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 hierarchy, use the −1 option. For example,

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

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

SEE ALSO

X(7), xrdb(1), listres(1)

AUTHOR

Jim Fulton, MIT X Consortium

NAME

bdftopcf − convert X font from Bitmap Distribution Format to Portable Compiled Format

SYNOPSIS

bdftopcf [**−p***n*] [**−u***n*] [**−m**] [**−l**] [**−M**] [**−L**] [**−t**] [**−i**] [**−o** *outputfile*] fontfile.bdf

DESCRIPTION

Bdftopcf is a font compiler for the X server and font server. 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(7)

AUTHOR

Keith Packard, MIT X Consortium

NAME

bdftruncate − generate truncated BDF font from ISO 10646-1-encoded BDF font

SYNOPSIS

bdftruncate *threshold* **<** *source.bdf* **>** *destination.bdf*

DESCRIPTION

bdftruncate allows one to generate from an ISO10646-1 encoded BDF font other ISO10646-1 BDF fonts in which all characters above a threshold code value are stored unencoded. This is often desirable because the Xlib API and X11 protocol data structures used for representing font metric information are extremely inefficient when handling sparsely populated fonts.

EXAMPLE

The command

```
bdftruncate 0x3200 < 6x13.bdf > 6x13t.bdf
```

will generate the file 6x13t.bdf in which all glyphs with codes >= 0x3200 will only be stored unencoded (i.e., they are encoded at codepoint -1).

SEE ALSO

ucs2any(1)

AUTHOR

bdftruncate was written by Markus Kuhn.

Branden Robinson wrote this manual page, originally for the Debian Project.

NAME

beforelight − screen saver

SYNOPSIS

beforelight [−*toolkitoption* . . .]

DESCRIPTION

The *beforelight* program is a sample implementation of a screen saver for X servers supporting the MIT-SCREEN-SAVER extension.

AUTHORS

Keith Packard, MIT X Consortium.

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 H 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/X11R6/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 ("")

AUTHOR

Davor Matic, MIT X Consortium

NAME

ccmakedep – create dependencies in makefiles using a C compiler

SYNOPSIS

ccmakedep [**cpp-flags**] [**–w***width*] [**–s***magic-string*] [**–f***makefile*] [**–o***object-suffix*] [**–v**] [**–a**] [**–cc***compiler*] [**––** *options* **––**] *sourcefile* . . .

DESCRIPTION

The **ccmakedep** program calls a C compiler to preprocess each *sourcefile*, and uses the output to construct *makefile* rules describing their dependencies. These rules instruct **make**(1) on which object files must be recompiled when a dependency has changed.

By default, **ccmakedep** 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 a line beginning with

```
# DO NOT DELETE
```

or one provided with the **–s** option, as a delimiter for the dependency output. If it finds it, it will delete everything following this up 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 makefile and place the output after that.

EXAMPLE

Normally, **ccmakedep** 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:
      ccmakedep −− $(CFLAGS) −− $(SRCS)
```

OPTIONS

The program will ignore any option that it does not understand, so you may use the same arguments that you would for **cc**(1), including **–D** and **–U** options to define and undefine symbols and **–I** to set the include path.

–a Append the dependencies to the file instead of replacing existing dependencies.

–cc*compiler*
: Use this compiler to generate dependencies.

–f*makefile*
: Filename. This allows you to specify an alternate makefile in which **ccmakedep** can place its output. Specifying "–" as the file name (that is, **–f–**) sends the output to standard output instead of modifying an existing file.

–s*string*
: Starting string delimiter. This option permits you to specify a different string for **ccmakedep** to look for in the makefile. The default is "# DO NOT DELETE".

–v Be verbose: display the C compiler command before running it.

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

SEE ALSO

cc(1), **make**(1), **makedepend**(1), **ccmakedep**(1).

AUTHOR

ccmakedep was written by the X Consortium.

Colin Watson wrote this manual page, originally for the Debian Project, based partly on the manual page for **makedepend**(1).

NAME

cleanlinks – remove dangling symbolic links and empty directories

SYNOPSIS

cleanlinks

DESCRIPTION

The *cleanlinks* program searches the directory tree descended from the current directory for symbolic links whose targets do not exist, and removes them. It then removes all empty directories in that directory tree.

cleanlinks is useful for cleaning up a shadow link tree created with **lndir**(1) after files have been removed from the real directory.

DIAGNOSTICS

A message will be printed upon encountering each dangling symlink and empty directory.

SEE ALSO

lndir(1).

AUTHOR

David Dawes wrote the *cleanlinks* program for XFree86.

Colin Watson wrote this manual page, originally for the Debian Project.

NAME

constype − print type of Sun console

SYNOPSIS

constype [*device_name*] [**−num**]

DESCRIPTION

The *constype* program writes on the standard output the Sun code for the type of display that the console is. The types output include these:

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.

The *device_name* argument, if given, is the device to examine. If not given, */dev/fb* is used.

The **−num** option causes *constype* to follow the type keyword with the numeric value of that type, as returned by the FBIOGATTR or FBIOGTYPE ioctl and defined in fbio.h. This is useful if the type is not recognized by the program.

EXIT STATUS

The program exits with status 0 if it identified a known console type, 1 if the type was unknown, and 2 if the device could not be opened or another error occurred.

BUGS

Not tested on all monitor types

COPYRIGHT

AUTHOR

Doug Moran <moran@ai.sri.com>

NAME

cxpm − Check an XPM (X PixMap) file - XPM 1, 2, or 3.

SYNOPSIS

cxpm [*filename*]

DESCRIPTION

The **cxpm** program can be used to check the format of any XPM (version 1, 2, or 3) file. On error, unlike **sxpm**, **cxpm** prints out an error message indicating where the parser choked. This should help finding out what's wrong with an XPM file but do not expect too much from it though. This is not even close from being some kind of lint program for XPM. First, it stops at the first error it encounters - so several fix and retry cycles may be necessary to get your file to parse successfully. Second, **cxpm** only cares about the format. If, for instance, your pixmap uses too many colors for your system you still may experience difficulties displaying it. Be warned.

When no *filename* is given **cxpm** reads from the standard input.

AUTHOR

Arnaud Le Hors (lehors@sophia.inria.fr)

NAME

dga − test program for the XFree86-DGA extension

SYNOPSIS

dga

DESCRIPTION

Dga is a simple test client for the XFree86-DGA extension. It fills the screen with a different colour for each key pressed. It prints some basic framebuffer parameters, and also keyboard and pointer events to stdout. To exit, hit the ‘q’ key. Hitting the ‘b’ key runs a simple benchmark, measuring raw framebuffer write and read speed (this takes one second each).

AUTHOR

Jon Tombs

NAME

dpsexec − Display PostScript Executive

SYNOPSIS

dpsexec [**−display** *name*][**−sync**][**−backup**][**−noexec**][**−root**][**−drawable** *windowId*][**−height** *n*][**−width** *n*]

DESCRIPTION

dpsexec is a Display PostScript program that allows the user to interact directly with the PostScript interpreter through a command interface. **dpsexec** reads lines of text from standard input and passes each line to the PostScript interpreter for execution. It creates a window that displays the results of graphics operations as they are executed. **dpsexec** exits when end of file is reached on standard input, or when the user types "quit<return>", which executes the PostScript **quit** operator.

By default, **dpsexec** executes the PostScript **executive** operator before it accepts any user input. This operator puts the PostScript interpreter in "interactive executive" mode so that the user can control the interpreter directly. In this mode, the PostScript interpreter supports certain line-editing functions and prompts the user when it is ready to execute more input. See section 2.4.4, "Using the Interpreter Interactively," of the *PostScript Language Reference Manual, Second Edition,* for detailed information on this mode of operation.

OPTIONS

−display *name*
: specifies the display on which to open a connection to the Display PostScript system. If no display is specified, the DISPLAY environment variable is used.

−sync
: establishes a synchronous connection with the specified X display.

−backup
: uses backing store for the window in which graphics are displayed, if possible. This is generally only effective with the DPS NX system.

−noexec
: prevents **dpsexec** from entering "interactive executive" mode. The primary effect of this option is to inhibit printing the **PS>** prompt before each line of input is accepted. This option is useful when **dpsexec** is run with standard input redirected from a file or a pipe.

−root
: tells **dpsexec** to draw into the root window instead of into a window that it creates.

−drawable *windowId*
: tells **dpsexec** to draw into the specified window instead of into a window that it creates.

−height *n*
: sets the height of the created window.

−width *n*
: sets the width of the created window.

DIAGNOSTICS

PostScript language error messages are printed to standard output.

AUTHOR

Adobe Systems Incorporated

NOTES

PostScript and Display PostScript are trademarks of Adobe Systems Incorporated which may be registered in certain jurisdictions.

NAME

dpsinfo − Display PostScript Information

SYNOPSIS

dpsinfo [**−display** *name*] [**-debug**]

DESCRIPTION

Dpsinfo is a utility for displaying information about the DPS extension present in an X server or provided by a client-side DPS agent. **Dpsinfo** will print out the version of the DPS protocol used, the language level and version of the underlying PS interpreter, as well as the set of font formats supported.

If **−debug** is specified, **dpsinfo** will print out all the PS code sent to the server.

SEE ALSO

X(1), xdpyinfo(1), dpsexec(1)

AUTHOR

Juliusz Chroboczek

NOTES

PostScript and Display PostScript are trademarks of Adobe Systems Incorporated which may be registered in certain jurisdictions.

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(7)*). 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 application 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 application'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 application'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

Get Widget Tree

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

Refresh Current 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.

For 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, select the *cancel* button.

Show Resource Box

This command will popup a resource box for the current application. 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 corresponding 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 application.

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 area 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, described 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 pop down 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 applications to 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 applications that are linked against Xaw to be able to speak to the resource editor. While this provides great flexibility, and is a useful tool, it can quite easily be abused. It is therefore possible for any Xaw application 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 application wishes to impose on the editres protocol.

The accepted values are:

all
: Block all requests.

setValues
: Block all SetValues requests. As this is the only editres request that actually modifies the application, this is in effect stating that the application is read-only.

none
: Allow all editres requests.

Remember that these resources are set on any Xaw application, **not editres**. They allow individual applications to keep all or some of the requests editres makes from ever succeeding. Of course, editres is also an Xaw application, 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 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 application 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 application>
                .
                .
                .
```

```
                    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/X11R6/lib/X11/app-defaults/Editres

specifies required resources

SEE ALSO

X(7), 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.

AUTHOR

Chris D. Peterson, formerly MIT X Consortium

NAME

fc-cache, fonts.cache – create an index of FreeType font files in a directory

SYNOPSIS

fc-cache [*directory-name* ...]

DESCRIPTION

If directory arguments are not given, *fc-cache* uses each directory in the current font configuration. Each directory is scanned for font files readable by FreeType. A cache is created which contains properties of each font and the associated filename. This cache is used to speed application startup when using the fontconfig library.

FILES

fonts.cache Maps file names to font properties. Read by the fontconfig library at application startup to locate appropriate fonts.

SEE ALSO

fontconfig(3)

NAME

fc-lang, fclang.h – create an database of language orthographies

SYNOPSIS

fc-lang [*language-coverage* . . .]

DESCRIPTION

Fc-lang builds the fclang.h file used in the fontconfig library to automatically determine language coverage for fonts which don't contain this information.

FILES

fclang.tmpl.h The template file in which the tables are inserted

SEE ALSO

fontconfig(3)

NAME

fc-list − list available fonts

SYNOPSIS

fc-list [*font-pattern*]

DESCRIPTION

If font pattern is not given, *fc-list* lists all available faces and styles in the current font configuration.

SEE ALSO

fontconfig(3)

NAME

fontconfig − Font configuration and customization library

SYNOPSIS

#include <fontconfig/fontconfig.h>
#include <fontconfig/fcfreetype.h>

DESCRIPTION

Fontconfig is a library designed to provide system-wide font configuration, customization and application access.

FUNCTIONAL OVERVIEW

Fontconfig contains two essential modules, the configuration module which builds an internal configuration from XML files and the matching module which accepts font patterns and returns the nearest matching font.

FONT CONFIGURATION

The configuration module consists of the FcConfig datatype, libexpat and FcConfigParse which walks over an XML tree and ammends a configuration with data found within. From an external perspective, configuration of the library consists of generating a valid XML tree and feeding that to FcConfigParse. The only other mechanism provided to applications for changing the running configuration is to add fonts and directories to the list of application-provided font files.

The intent is to make font configurations relatively static, and shared by as many applications as possible. It is hoped that this will lead to more stable font selection when passing names from one application to another. XML was chosen as a configuration file format because it provides a format which is easy for external agents to edit while retaining the correct structure and syntax.

Font configuration is separate from font matching; applications needing to do their own matching can access the available fonts from the library and perform private matching. The intent is to permit applications to pick and choose appropriate functionality from the library instead of forcing them to choose between this library and a private configuration mechanism. The hope is that this will ensure that configuration of fonts for all applications can be centralized in one place. Centralizing font configuration will make simplify and regularize font installation and customization.

FONT PROPERTIES

While font patterns may contain essentially any properties, there are some well known properties with associated types. Fontconfig uses some of these properties for font matching and font completion. Others are provided as a convenience for the applications rendering mechanism.

Property	CPP symbol	Type	Description
family	FC_FAMILY	String	Font family name
style	FC_STYLE	String	Font style. Overrides weight and slant
slant	FC_SLANT	Int	Italic, oblique or roman
weight	FC_WEIGHT	Int	Light, medium, demibold, bold or black
size	FC_SIZE	Double	Point size
aspect	FC_ASPECT	Double	Stretches glyphs horizontally before hinting
pixelsize	FC_PIXEL_SIZE	Double	Pixel size
spacing	FC_SPACING	Int	Proportional, monospace or charcell
foundry	FC_FOUNDRY	String	Font foundry name
antialias	FC_ANTIALIAS	Bool	Whether glyphs can be antialiased
hinting	FC_HINTING	Bool	Whether the rasterizer should use hinting
verticallayout	FC_VERTICAL_LAYOUT	Bool	Use vertical layout
autohint	FC_AUTOHINT	Bool	Use autohinter instead of normal hinter

globaladvance	FC_GLOBAL_ADVANCE	Bool	Use font global advance data
file	FC_FILE	String	The filename holding the font
index	FC_INDEX	Int	The index of the font within the file
ftface	FC_FT_FACE	FT_Face	Use the specified FreeType face object
rasterizer	FC_RASTERIZER	String	Which rasterizer is in use
outline	FC_OUTLINE	Bool	Whether the glyphs are outlines
scalable	FC_SCALABLE	Bool	Whether glyphs can be scaled
scale	FC_SCALE	Double	Scale factor for point->pixel conversions
dpi	FC_DPI	Double	Target dots per inch
rgba	FC_RGBA	Int	unknown, rgb, bgr, vrgb, vbgr, none - subpixel geometry
minspace	FC_MINSPACE	Bool	Eliminate leading from line spacing
charset	FC_CHARSET	CharSet	Unicode chars encoded by the font
lang	FC_LANG	String	List of RFC-3066-style languages this font supports

FONT MATCHING

Fontconfig performs matching by measuring the distance from a provided pattern to all of the available fonts in the system. The closest matching font is selected. This ensures that a font will always be returned, but doesn't ensure that it is anything like the requested pattern.

Font matching starts with an application constructed pattern. The desired attributes of the resulting font are collected together in an FcPattern object. Each property of the pattern can contain one or more values; these are listed in priority order; matches earlier in the list are considered "closer" than matches later in the list.

The initial pattern is modified by applying the list of editing instructions specific to patterns found in the configuration; each consists of a match predicate and a set of editing operations. They are executed in the order they appeared in the configuration. Each match causes the associated sequence of editing operations to be applied.

After the pattern has been edited, a sequence of default substitutions are performed to canonicalize the set of available properties; this avoids the need for the lower layers to constantly provide default values for various font properties during rendering.

The canonical font pattern is finally matched against all available fonts. The distance from the pattern to the font is measured for each of several properties: foundry, charset, family, lang, spacing, pixelsize, style, slant, weight, antialias, rasterizer and outline. This list is in priority order -- results of comparing earlier elements of this list weigh more heavily than later elements.

There is one special case to this rule; family names are split into two bindings; strong and weak. Strong family names are given greater precedence in the match than lang elements while weak family names are given lower precedence than lang elements. This permits the document language to drive font selection when any document specified font is unavailable.

The pattern representing that font is augmented to include any properties found in the pattern but not found in the font itself; this permits the application to pass rendering instructions or any other data through the matching system. Finally, the list of editing instructions specific to fonts found in the configuration are applied to the pattern. This modified pattern is returned to the application.

The return value contains sufficient information to locate and rasterize the font, including the file name, pixel size and other rendering data. As none of the information involved pertains to the FreeType library, applications are free to use any rasterization engine or even to take the identified font file and access it directly.

The match/edit sequences in the configuration are performed in two passes because there are essentially two different operations necessary -- the first is to modify how fonts are selected; aliasing families and adding suitable defaults. The second is to modify how the selected fonts are rasterized. Those must apply to the selected font, not the original pattern as false matches will often occur.

FONT LIST MATCHING

While many applications want to locate a single font best matching their search criteria, other applications need to build a set of fonts which can be used to present any Unicode data. Fontconfig provides an API to generate a list sorted by the nearness of each font to the pattern. Every font in the system is considered, the best matching fonts are placed first. The application then can select whether the remaining fonts are unconditionally included in the list, or whether they are included only if they cover portions of Unicode not covered by any of the preceeding fonts.

The list resulting from this match is represented by references to the original font patterns and so consumes very little memory. Using a list entry involves creating a pattern which combines the information from the font with the information from the original pattern and executing the font substitutions.

FONT NAMES

Fontconfig provides a textual representation for patterns that the library can both accept and generate. The representation is in three parts, first a list of family names, second a list of point sizes and finally a list of additional properties:

<families>-<point sizes>:<name1>=<values1>:<name2>=<values2>...

Values in a list are separated with commas. The name needn't include either families or point sizes; they can be elided. In addition, there are symbolic constants that simultaneously indicate both a name and a value. Here are some examples:

Times-12	12 point Times Roman
Times-12:bold	12 point Times Bold
Courier:italic	Courier Italic in the default size
Monospace:matrix=1 .1 0 1	The users preferred monospace font with artificial obliquing

LANG TAGS

Each font in the database contains a list of languages it supports. This is computed by comparing the Unicode coverage of the font with the orthography of each language. Languages are tagged using an RFC-3066 compatible naming and occur in two parts -- the ISO639 language tag followed a hyphen and then by the ISO 3166 country code. The hyphen and country code may be elided.

Fontconfig has orthographies for several languages built into the library. No provision has been made for adding new ones aside from rebuilding the library. It currently supports 122 of the 139 languages named in ISO 639-1, 141 of the languages with two-letter codes from ISO 639-2 and another 30 languages with only three-letter codes.

DATATYPES

FcChar8
FcChar16
FcChar32
FcBool These are primitive datatypes; the FcChar* types hold precisely the number of bits stated (if supported by the C implementation). FcBool holds one of two CPP symbols: FcFalse or FcTrue.

FcMatrix

An FcMatrix holds an affine transformation, usually used to reshape glyphs. A small set of matrix operations are provided to manipulate these.

```
typedef struct _FcMatrix {
        double xx, xy, yx, yy;
} FcMatrix;
```

FcCharSet

An FcCharSet is an abstract type that holds the set of encoded unicode chars in a font. Operations to build and compare these sets are provided.

FcType

Tags the kind of data stored in an FcValue.

FcValue

An FcValue object holds a single value with one of a number of different types. The 'type' tag indicates which member is valid.

```
typedef struct _FcValue {
        FcType type;
        union {
                const FcChar8 *s;
                int i;
                FcBool b;
                double d;
                const FcMatrix *m;
                const FcCharSet *c;
        } u;
} FcValue;
```

type	Union member	Datatype
FcTypeVoid	(none)	(none)
FcTypeInteger	i	int
FcTypeDouble	d	double
FcTypeString	s	char *
FcTypeBool	b	b
FcTypeMatrix	m	FcMatrix *
FcTypeCharSet	c	FcCharSet *

FcPattern

holds a set of names with associated value lists; each name refers to a property of a font. FcPatterns are used as inputs to the matching code as well as holding information about specific fonts. Each property can hold one or more values; conventionally all of the same type, although the interface doesn't demand that.

FcFontSet

```
typedef struct _FcFontSet {
        int nfont;
        int sfont;
        FcPattern **fonts;
} FcFontSet;
```

An FcFontSet contains a list of FcPatterns. Internally fontconfig uses this data structure to hold sets of fonts. Externally, fontconfig returns the results of listing fonts in this format. 'nfont' holds the number of patterns in the 'fonts' array; 'sfont' is used to indicate the size of that array.

FcStrSet

FcStrList FcStrSet holds a list of strings that can be appended to and enumerated. Its unique characteristic is that the enumeration works even while strings are appended during enumeration. FcStrList is used during enumeration to safely and correctly walk the list of strings even while that list is edited in the middle of enumeration.

FcObjectSet

```
typedef struct _FcObjectSet {
    int nobject;
    int sobject;
    const char **objects;
} FcObjectSet;
```

holds a set of names and is used to specify which fields from fonts are placed in the the list of returned patterns when listing fonts.

FcObjectType

typedef struct _FcObjectType { const char *object; FcType type; } FcObjectType;

marks the type of a pattern element generated when parsing font names. Applications can add new object types so that font names may contain the new elements.

FcConstant

typedef struct _FcConstant { const FcChar8 *name; const char *object; int value; } FcConstant;

Provides for symbolic constants for new pattern elements. When 'name' is seen in a font name, an 'object' element is created with value 'value'.

FcBlanks

holds a list of Unicode chars which are expected to be blank; unexpectedly blank chars are assumed to be invalid and are elided from the charset associated with the font.

FcFileCache

holds the per-user cache information for use while loading the font database. This is built automatically for the current configuration when that is loaded. Applications must always pass '0' when one is requested.

FcConfig

holds a complete configuration of the library; there is one default configuration, other can be constructed from XML data structures. All public entry points that need global data can take an optional FcConfig* argument; passing 0 uses the default configuration. FcConfig objects hold two sets of fonts, the first contains those specified by the configuration, the second set holds those added by the application at run-time. Interfaces that need to reference a particulat set use one of the FcSetName enumerated values.

FcSetName

Specifies one of the two sets of fonts available in a configuration; FcSetSystem for those fonts specified in the configuration and FcSetApplication which holds fonts provided by the application.

FcResult

Used as a return type for functions manipulating FcPattern objects.

Result code	Meaning

FcResultMatch	Object exists with the specified ID
FcResultNoMatch	Object doesn't exist at all
FcResultTypeMismatch	Object exists, but the type doesn't match
FcResultNoId	Object exists, but has fewer values than specified

FcAtomic

Used for locking access to config files. Provides a safe way to update configuration files.

FUNCTIONS

FcMatrix

FcMatrix structures hold an affine transformation in matrix form.

Initializes a matrix to the identify transformation.

FcMatrix *FcMatrixCopy (const FcMatrix *mat)

Allocates a new FcMatrix and copies 'mat' into it.

FcBool FcMatrixEqual (const FcMatrix *mat1, const FcMatrix *mat2)

Returns FcTrue if 'mat1' and 'mat2' are equal, else FcFalse.

void FcMatrixMultiply (FcMatrix *result, const FcMatrix *a, const FcMatrix *b)

Multiplies 'a' and 'b' together, placing the result in 'result'. 'result' may refer to the sam matrix as either 'a' or 'b'.

void FcMatrixRotate (FcMatrix *m, double c, double s)

If 'c' is cos(angle) and 's' is sin(angle), FcMatrixRotate rotates the matrix by 'angle'.

void FcMatrixScale (FcMatrix *m, double sx, double sy)

Scales 'm' by 'sx' in the horizontal dimension and 'sy' in the vertical dimension.

void FcMatrixShear (FcMatrix *m, double sh, double sv)

Shears 'm' by 'sh' in the horizontal direction and 'sv' in the vertical direction.

FcCharSet

An FcCharSet is a boolean array indicating a set of unicode chars. Those associated with a font are marked constant and cannot be edited. FcCharSets may be reference counted internally to reduce memory consumption; this may be visible to applications as the result of FcCharSetCopy may return it's argument, and that CharSet may remain unmodifiable.

FcCharSet *FcCharSetCreate (void)

Creates an empty FcCharSet object.

void FcCharSetDestroy (FcCharSet *fcs)

Frees an FcCharSet object.

FcBool FcCharSetAddChar (FcCharSet *fcs, FcChar32 ucs4)

Adds a single unicode char to the set, returning FcFalse on failure, either as a result of a constant set or from running out of memory.

FcCharSet *FcCharSetCopy (FcCharSet *src)
Makes a copy of 'src'; note that this may not actually do anything more than increment the reference count on 'src'.

FcBool FcCharSetEqual (const FcCharSet *a, const FcCharSet *b)
Returns whether 'a' and 'b' contain the same set of unicode chars.

FcCharSet *FcCharSetIntersect (const FcCharSet *a, const FcCharSet *b)
Returns a set including only those chars found in both 'a' and 'b'.

FcCharSet *FcCharSetUnion (const FcCharSet *a, const FcCharSet *b);
Returns a set including only those chars found in either 'a' or 'b'.

FcCharSet *FcCharSetSubtract (const FcCharSet *a, const FcCharSet *b)
Returns a set including only those chars found in 'a' but not 'b'.

FcBool FcCharSetHasChar (const FcCharSet *fcs, FcChar32 ucs4)
Returns whether 'fcs' contains the char 'ucs4'.

FcChar32 FcCharSetCount (const FcCharSet *a)
Returns the total number of unicode chars in 'a'.

FcChar32 FcCharSetIntersectCount (const FcCharSet *a, const FcCharSet *b)
Returns the number of chars that are in both 'a' and 'b'.

FcChar32 FcCharSetSubtractCount (const FcCharSet *a, const FcCharSet *b)
Returns the number of chars that are in 'a' but not in 'b'.

FcBool FcCharSetIsSubset (const FcCharSet *a, const FcCharSet *b)
Returns whether 'a' is a subset of 'b'.

FcChar32 FcCharSetFirstPage (const FcCharSet *a, FcChar32 [FC_CHARSET_MAP_SIZE], FcChar32 *next)
Builds an array of bits marking the first page of Unicode coverage of 'a'. Returns the base of the array. 'next' contains the next page in the font.

FcChar32 FcCharSetNextPage (const FcCharSet *a, FcChar32 [FC_CHARSET_MAP_SIZE], FcChar32 *next)
Builds an array of bits marking the Unicode coverage of 'a' for page '*next'. Returns the base of the array. 'next' contains the next page in the font.

FcValue

FcValue is a structure containing a type tag and a union of all possible datatypes. The tag is an enum of type **FcType** and is intended to provide a measure of run-time typechecking, although that depends on careful programming.

void FcValueDestroy (FcValue v)
Frees any memory referenced by 'v'. Values of type FcTypeString, FcTypeMatrix and FcTypeCharSet reference memory, the other types do not.

FcValue FcValueSave (FcValue v)
Returns a copy of ‘v’ duplicating any object referenced by it so that ‘v’ may be safely destroyed without harming the new value.

FcPattern

An FcPattern is an opaque type that holds both patterns to match against the available fonts, as well as the information about each font.

FcPattern *FcPatternCreate (void)
Creates a pattern with no properties; used to build patterns from scratch.

void FcPatternDestroy (FcPattern *p)
Destroys a pattern, in the process destroying all related values.

FcBool FcPatternEqual (const FcPattern *pa, const FcPattern *pb);
Returns whether ’pa’ and ’pb’ are exactly alike.

FcBool FcPatternEqualSubset (const FcPattern *pa, const FcPattern *pb, const FcObjectSet *os)
Returns whether ’pa’ and ’pb’ have exactly the same values for all of the objects in ’os’.

FcChar32 FcPatternHash (const FcPattern *p)
Returns a 32-bit number which is the same for any two patterns which are exactly alike.

FcBool FcPatternAdd (FcPattern *p, const char *object, FcValue value, FcBool append)
Adds a single value to the list of values associated with the property named ‘object’. If ‘append’ is FcTrue, the value is added at the end of any existing list, otherwise it is inserted at the begining. ‘value’ is saved (with FcValueSave) when inserted into the pattern so that the library retains no reference to any application-supplied data structure.

FcBool FcPatternAddWeak (FcPattern *p, const char *object, FcValue value, FcBool append)
FcPatternAddWeak is essentially the same as FcPatternAdd except that any values added to the list have binding ’weak’ instead of ’strong’.

FcBool FcPatternAddInteger (FcPattern *p, const char *object, int i)
FcBool FcPatternAddDouble (FcPattern *p, const char *object, double d)
FcBool FcPatternAddString (FcPattern *p, const char *object, const char *s)
FcBool FcPatternAddMatrix (FcPattern *p, const char *object, const FcMatrix *s)
FcBool FcPatternAddCharSet (FcPattern *p, const char *object, const FcCharSet *c)
FcBool FcPatternAddBool (FcPattern *p, const char *object, FcBool b)
These are all convenience functions that insert objects of the specified type into the pattern. Use these in preference to FcPatternAdd as they will provide compile-time typechecking. These all append values to any existing list of values.

FcResult FcPatternGet (FcPattern *p, const char *object, int id, FcValue *v)
Returns in ‘v’ the ‘id’th value associated with the property ‘object’. The value returned is not a copy, but rather refers to the data stored within the pattern directly. Applications must not free this value.

FcResult FcPatternGetInteger (FcPattern *p, const char *object, int n, int *i);
FcResult FcPatternGetDouble (FcPattern *p, const char *object, int n, double *d);
FcResult FcPatternGetString (FcPattern *p, const char *object, int n, char **const s);
FcResult FcPatternGetMatrix (FcPattern *p, const char *object, int n, FcMatrix **s);
FcResult FcPatternGetCharSet (FcPattern *p, const char *object, int n, FcCharSet **c);
FcResult FcPatternGetBool (FcPattern *p, const char *object, int n, FcBool *b);

These are convenience functions that call FcPatternGet and verify that the returned data is of the expected type. They return FcResultTypeMismatch if this is not the case. Note that these (like FcPatternGet) do not make a copy of any data structure referenced by the return value. Use these in preference to FcPatternGet to provide compile-time typechecking.

FcPattern *FcPatternBuild (FcPattern *orig, ...);
FcPattern *FcPatternVaBuild (FcPattern *orig, va_list va)

Builds a pattern using a list of objects, types and values. Each value to be entered in the pattern is specified with three arguments:

1. Object name, a string describing the property to be added.

2. Object type, one of the FcType enumerated values

3. Value, not an FcValue, but the raw type as passed to any of the FcPatternAdd<type> functions. Must match the type of the second argument.

The argument list is terminated by a null object name, no object type nor value need be passed for this. The values are added to ‘pattern’, if ‘pattern’ is null, a new pattern is created. In either case, the pattern is returned. Example:

pattern = FcPatternBuild (0, FC_FAMILY, FtTypeString, "Times", (char *) 0);

FcPatternVaBuild is used when the arguments are already in the form of a varargs value.

FcBool FcPatternDel (FcPattern *p, const char *object)

Deletes all values associated with the property ‘object’, returning whether the property existed or not.

void FcPatternPrint (const FcPattern *p)

Prints an easily readable version of the pattern to stdout. There is no provision for reparsing data in this format, it’s just for diagnostics and debugging.

void FcDefaultSubstitute (FcPattern *pattern)

Supplies default values for underspecified font patterns:

- Patterns without a specified style or weight are set to Medium
- Patterns without a specified style or slant are set to Roman
- Patterns without a specified pixel size are given one computed from any specified point size (default 12), dpi (default 75) and scale (default 1).

FcPattern *FcNameParse (const char *name)

Converts ’name’ from the standard text format described above into a pattern.

FcChar8 *FcNameUnparse (FcPattern *pat)

Converts the given pattern into the standard text format described above. The return value is not static, but instead refers to newly allocated memory which should be freed by the caller.

FcFontSet

An FcFontSet simply holds a list of patterns; these are used to return the results of listing available fonts.

FcFontSet *FcFontSetCreate (void)
: Creates an empty font set.

void FcFontSetDestroy (FcFontSet *s);
: Destroys a font set. Note that this destroys any referenced patterns as well.

FcBool FcFontSetAdd (FcFontSet *s, FcPattern *font)
: Adds a pattern to a font set. Note that the pattern is not copied before being inserted into the set.

FcObjectSet

An FcObjectSet holds a list of pattern property names; it is used to indiciate which properties are to be returned in the patterns from FcFontList.

FcObjectSet *FcObjectSetCreate (void)
: Creates an empty set.

FcBool FcObjectSetAdd (FcObjectSet *os, const char *object)
: Adds a proprety name to the set.

void FcObjectSetDestroy (FcObjectSet *os)
: Destroys an object set.

FcObjectSet *FcObjectSetBuild (const char *first, ...)
FcObjectSet *FcObjectSetVaBuild (const char *first, va_list va)
: These build an object set from a null-terminated list of property names.

FcObjectType

Provides for applcation-specified font name object types so that new pattern elements can be generated from font names.

FcBool FcNameRegisterObjectTypes (const FcObjectType *types, int ntype)
: Register ’ntype’ new object types.

FcBool FcNameUnregisterObjectTypes (const FcObjectType *types, int ntype)
: Unregister ’ntype’ object types.

const FcObjectType *FcNameGetObjectType (const char *object)
: Return the object type for the pattern element named ’object’.

FcConstant

Provides for application-specified symbolic constants for font names.

FcBool FcNameRegisterConstants (const FcConstant *consts, int nconsts)
: Register ’nconsts’ new symbolic constants.

FcBool FcNameUnregisterConstants (const FcConstant *consts, int nconsts)
Unregister 'nconsts' symbolic constants.

const FcConstant *FcNameGetConstant (FcChar8 *string)
Return the FcConstant structure related to symbolic constant 'string'.

FcBool FcNameConstant (FcChar8 *string, int *result);
Returns whether a symbolic constant with name 'string' is registered, placing the value of the constant in 'result' if present.

FcBlanks

An FcBlanks object holds a list of Unicode chars which are expected to be blank when drawn. When scanning new fonts, any glyphs which are empty and not in this list will be assumed to be broken and not placed in the FcCharSet associated with the font. This provides a significantly more accurate CharSet for applications.

FcBlanks *FcBlanksCreate (void)
Creates an empty FcBlanks object.

void FcBlanksDestroy (FcBlanks *b)
Destroys an FcBlanks object, freeing any associated memory.

FcBool FcBlanksAdd (FcBlanks *b, FcChar32 ucs4)
Adds a single character to an FcBlanks object, returning FcFalse if this process ran out of memory.

FcBool FcBlanksIsMember (FcBlanks *b, FcChar32 ucs4)
Returns whether the specified FcBlanks object contains the indicated Unicode value.

FcConfig

An FcConfig object holds the internal representation of a configuration. There is a default configuration which applications may use by passing 0 to any function using the data within an FcConfig.

FcConfig *FcConfigCreate (void)
Creates an empty configuration.

void FcConfigDestroy (FcConfig *config)
Destroys a configuration and any data associated with it. Note that calling this function with the return from FcConfigGetCurrent will place the library in an indeterminate state.

FcBool FcConfigSetCurrent (FcConfig *config)
Sets the current default configuration to 'config'. Implicitly calls FcConfigBuildFonts if necessary, returning FcFalse if that call fails.

FcConfig *FcConfigGetCurrent (void)
Returns the current default configuration.

FcBool FcConfigUptoDate (FcConfig *config)
Checks all of the files related to 'config' and returns whether the in-memory version is in sync with the disk version.

FcBool FcConfigBuildFonts (FcConfig *config)
Builds the set of available fonts for the given configuration. Note that any changes to the configuration after this call have indeterminate effects. Returns FcFalse if this operation runs out of memory.

FcStrList *FcConfigGetConfigDirs (FcConfig *config)
Returns the list of font directories specified in the configuration files for 'config'. Does not include any subdirectories.

FcStrList *FcConfigGetFontDirs (FcConfig *config)
Returns the list of font directories in 'config'. This includes the configured font directories along with any directories below those in the filesystem.

FcStrList *FcConfigGetConfigFiles (FcConfig *config)
Returns the list of known configuration files used to generate 'config'. Note that this will not include any configuration done with FcConfigParse.

char *FcConfigGetCache (FcConfig *config)
Returns the name of the file used to store per-user font information.

FcFontSet *FcConfigGetFonts (FcConfig *config, FcSetName set)
Returns one of the two sets of fonts from the configuration as specified by 'set'.

FcBlanks *FcConfigGetBlanks (FcConfig *config)
Returns the FcBlanks object associated with the given configuration, if no blanks were present in the configuration, this function will return 0.

int FcConfigGetRescanInverval (FcConfig *config)
Returns the interval between automatic checks of the configuration (in seconds) specified in 'config'. The configuration is checked during a call to FcFontList when this interval has passed since the last check.

FcBool FcConfigSetRescanInverval (FcConfig *config, int rescanInterval)
Sets the rescan interval; returns FcFalse if an error occurred.

FcBool FcConfigAppFontAddFile (FcConfig *config, const char *file)
Adds an application-specific font to the configuration.

FcBool FcConfigAppFontAddDir (FcConfig *config, const char *dir)
Scans the specified directory for fonts, adding each one found to the application-specific set of fonts.

void FcConfigAppFontClear (FcConfig *config)
Clears the set of application-specific fonts.

FcBool FcConfigSubstituteWithPat (FcConfig *config, FcPattern *p, FcPattern *p_pat FcMatchKind kind)
Performs the sequence of pattern modification operations, if 'kind' is FcMatchPattern, then those tagged as pattern operations are applied, else if 'kind' is FcMatchFont, those tagged as font operations are applied and p_pat is used for <test> elements with target=pattern.

FcBool FcConfigSubstitute (FcConfig *config, FcPattern *p, FcMatchKind kind)
Calls FcConfigSubstituteWithPat setting p_pat to NULL.

FcPattern *FcFontMatch (FcConfig *config, FcPattern *p, FcResult *result)
Returns the font in 'config' most close matching 'p'. This function should be called only after FcConfigSubstitute and FcDefaultSubstitute have been called for 'p'; otherwise the results will not be correct.

FcFontSet *FcFontSort (FcConfig *config, FcPattern *p, FcBool trim, FcCharSet **csp, FcResult *result) Returns the list of fonts sorted by
closeness to 'p'. If 'trim' is FcTrue, elements in the list which don't include Unicode coverage not provided by earlier elements in the list are elided. The union of Unicode coverage of all of the fonts is returned in 'csp', if 'csp' is not NULL. This function should be called only after FcConfigSubstitute and FcDefaultSubstitute have been called for 'p'; otherwise the results will not be correct.

The returned FcFontSet references FcPattern structures which may be shared by the return value from multiple FcFontSort calls, applications must not modify these patterns. Instead, they should be passed, along with 'p' to FcFontRenderPrepare which combines them into a complete pattern.

The FcFontSet returned by FcFontSort is destroyed by caling FcFontSetDestroy.

FcPattern *FcFontRenderPrepare (FcConfig *config, FcPattern *pat, FcPattern *font)
Creates a new pattern consisting of elements of 'font' not appearing in 'pat', elements of 'pat' not appearing in 'font' and the best matching value from 'pat' for elements appearing in both. The result is passed to FcConfigSubstitute with 'kind' FcMatchFont and then returned.

FcFontSet *FcFontList (FcConfig *config, FcPattern *p, FcObjectSet *os)
Selects fonts matching 'p', creates patterns from those fonts containing only the objects in 'os' and returns the set of unique such patterns.

char *FcConfigFilename (const char *name)
Given the specified external entity name, return the associated filename. This provides applications a way to convert various configuration file references into filename form.

A null or empty 'name' indicates that the default configuration file should be used; which file this references can be overridden with the FC_CONFIG_FILE environment variable. Next, if the name starts with '~', it refers to a file in the current users home directory. Otherwise if the name doesn't start with '/', it refers to a file in the default configuration directory; the built-in default directory can be overridden with the FC_CONFIG_DIR environment variable.

Initialization

These functions provide some control over how the library is initialized.

FcConfig *FcInitLoadConfig (void)
Loads the default configuration file and returns the resulting configuration. Does not load any font information.

FcConfig *FcInitLoadConfigAndFonts (void)
Loads the default configuration file and builds information about the available fonts. Returns the resulting configuration.

FcBool FcInit (void)
Loads the default configuration file and the fonts referenced therein and sets the default configuration to that result. Returns whether this process succeeded or not. If the default configuration has already been loaded, this routine does nothing and returns FcTrue.

int FcGetVersion (void)
Returns the version number of the library.

FcBool FcInitReinitialize (void)
Forces the default configuration file to be reloaded and resets the default configuration.

FcBool FcInitBringUptoDate (void)
Checks the rescan interval in the default configuration, checking the configuration if the interval has passed and reloading the configuration if when any changes are detected.

FcAtomic

These functions provide a safe way to update config files, allowing ongoing reading of the old config file while locked for writing and ensuring that a consistent and complete version of the config file is always available.

FcAtomic * FcAtomicCreate (const FcChar8 *file)
Creates a data structure containing data needed to control access to 'file'. Writing is done to a separate file. Once that file is complete, the original configuration file is atomically replaced so that reading process always see a consistent and complete file without the need to lock for reading.

FcBool FcAtomicLock (FcAtomic *atomic)
Attempts to lock the file referenced by 'atomic'. Returns FcFalse if the file is locked by another process, else returns FcTrue and leaves the file locked.

FcChar8 *FcAtomicNewFile (FcAtomic *atomic)
Returns the filename for writing a new version of the file referenced by 'atomic'.

FcChar8 *FcAtomicOrigFile (FcAtomic *atomic)
Returns the file refernced by 'atomic'.

FcBool FcAtomicReplaceOrig (FcAtomic *atomic)
Replaces the original file referenced by 'atomic' with the new file.

void FcAtomicDeleteNew (FcAtomic *atomic)
Deletes the new file.

void FcAtomicUnlock (FcAtomic *atomic)
Unlocks the file.

void FcAtomicDestroy (FcAtomic *atomic)
Destroys 'atomic'.

FreeType specific functions

#include <fontconfig/fcfreetype.h>

While the fontconfig library doesn't insist that FreeType be used as the rasterization mechanism for fonts, it does provide some convenience functions.

FT_UInt FcFreeTypeCharIndex (FT_Face face, FcChar32 ucs4)

Maps a Unicode char to a glyph index. This function uses information from several possible underlying encoding tables to work around broken fonts. As a result, this function isn't designed to be used in performance sensitive areas; results from this function are intended to be cached by higher level functions.

FcCharSet *FcFreeTypeCharSet (FT_Face face, FcBlanks *blanks) Scans a

FreeType face and returns the set of encoded Unicode chars. This scans several encoding tables to build as complete a list as possible. If 'blanks' is not 0, the glyphs in the font are examined and any blank glyphs not in 'blanks' are not placed in the returned FcCharSet.

FcPattern *FcFreeTypeQuery (const char *file, int id, FcBlanks *blanks, int *count)

Constructs a pattern representing the 'id'th font in 'file'. The number of fonts in 'file' is returned in 'count'.

XML specific functions

FcBool FcConfigParseAndLoad (FcConfig *config, const FcChar8 *file, FcBool complain)

Walks the configuration in 'file' and constructs the internal representation in 'config'. Any include files referenced from within 'file' will be loaded with FcConfigLoad and also parsed. If 'complain' is FcFalse, no warning will be displayed if 'file' does not exist.

File and Directory routines

FcBool FcFileScan (FcFontSet *set, FcStrSet *dirs, FcFileCache *cache, FcBlanks *blanks, const char *file, FcBool force)

Scans a single file and adds all fonts found to 'set'. If 'force' is FcTrue, then the file is scanned even if associated information is found in 'cache'. If 'file' is a directory, it is added to 'dirs'.

FcBool FcDirScan (FcFontSet *set, FcStrSet *dirs, FcFileCache *cache, FcBlanks *blanks, const char *dir, FcBool force)

Scans an entire directory and adds all fonts found to 'set'. If 'force' is FcTrue, then the directory and all files within it are scanned even if information is present in the per-directory cache file or 'cache'. Any subdirectories found are added to 'dirs'.

FcBool FcDirSave (FcFontSet *set, FcStrSet *dirs, const char *dir)

Creates the per-directory cache file for 'dir' and populates it with the fonts in 'set' and subdirectories in 'dirs'.

FcBool FcDirCacheValid (const FcChar8 *cache_file)

Returns FcTrue if 'cache_file' is no older than the directory containing it, else FcFalse.

FcStrSet and FcStrList

A data structure for enumerating strings, used to list directories while scanning the configuration as directories are added while scanning.

FcStrSet *FcStrSetCreate (void)
: Create an empty set.

FcBool FcStrSetMember (FcStrSet *set, const FcChar8 *s)
: Returns whether 's' is a member of 'set'.

FcBool FcStrSetAdd (FcStrSet *set, const FcChar8 *s)
: Adds a copy of 's' to 'set'.

FcBool FcStrSetAddFilename (FcStrSet *set, const FcChar8 *s)
: Adds a copy 's' to 'set', The copy is created with FcStrCopyFilename so that leading '~' values are replaced with the value of the HOME environment variable.

FcBool FcStrSetDel (FcStrSet *set, const FcChar8 *s)
: Removes 's' from 'set', returning FcTrue if 's' was a member else FcFalse.

void FcStrSetDestroy (FcStrSet *set)
: Destroys 'set'.

FcStrList *FcStrListCreate (FcStrSet *set)
: Creates an enumerator to list the strings in 'set'.

FcChar8 *FcStrListNext (FcStrList *list)
: Returns the next string in 'set'.

void FcStrListDone (FcStrList *list)
: Destroys the enumerator 'list'.

String utilities

int FcUtf8ToUcs4 (FcChar8 *src, FcChar32 *dst, int len)
: Converts the next Unicode char from 'src' into 'dst' and returns the number of bytes containing the char. 'src' nust be at least 'len' bytes long.

int FcUcs4ToUtf8 (FcChar32 src, FcChar8 dst[FC_UTF8_MAX_LEN])
: Converts the Unicode char from 'src' into 'dst' and returns the number of bytes needed to encode the char.

FcBool FcUtf8Len (FcChar8 *src, int len, int *nchar, int *wchar)
: Counts the number of Unicode chars in 'len' bytes of 'src'. Places that count in 'nchar'. 'wchar' contains 1, 2 or 4 depending on the number of bytes needed to hold the largest unicode char counted. The return value indicates whether 'src' is a well-formed UTF8 string.

int FcUtf16ToUcs4 (FcChar8 *src, FcEndian endian, FcChar32 *dst, int len)
: Converts the next Unicode char from 'src' into 'dst' and returns the number of bytes containing the char. 'src' must be at least 'len' bytes long. Bytes of 'src' are combined into 16-bit units according to 'endian'.

FcBool FcUtf16Len (FcChar8 *src, FcEndian endian, int len, int *nchar, int *wchar)
Counts the number of Unicode chars in 'len' bytes of 'src'. Bytes of 'src' are combined into 16-bit units according to 'endian'. Places that count in 'nchar'. 'wchar' contains 1, 2 or 4 depending on the number of bytes needed to hold the largest unicode char counted. The return value indicates whether 'string' is a well-formed UTF16 string.

FcChar8 *FcStrCopy (const FcChar8 *s)
Allocates memory, copies 's' and returns the resulting buffer. Yes, this is 'strdup', but that function isn't available on every platform.

FcChar8 *FcStrCopyFilename (const FcChar8 *s)
Just like FcStrCopy except that it converts any leading '~' characters in 's' to the value of the HOME environment variable.

int FcStrCmpIgnoreCase (const char *s1, const char *s2)
Returns the usual <0, 0, >0 result of comparing 's1' and 's2'. This test is case-insensitive in the ASCII range and will operate properly with UTF8 encoded strings, although it does not check for well formed strings.

FcChar8 *FcStrDirname (const FcChar8 *file)
Returns the directory containing 'file'.

FcChar8 *FcStrBasename (const FcChar8 *file)
Returns the filename of 'file' stripped of any leading directory names.

CONFIGURATION FILE FORMAT

Configuration files for fontconfig are stored in XML format; this format makes external configuration tools easier to write and ensures that they will generate syntactically correct configuration files. As XML files are plain text, they can also be manipulated by the expert user using a text editor.

The fontconfig document type definition resides in the external entity "fonts.dtd"; this is normally stored in the default font configuration directory (/etc/fonts). Each configuration file should contain the following structure:

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

<fontconfig>

This is the top level element for a font configuration and can contain <dir>, <cache>, <include>, <match> and <alias> elements in any order.

<dir>

This element contains a directory name which will be scanned for font files to include in the set of available fonts.

<cache>

This element contains a file name for the per-user cache of font information. If it starts with '~', it refers to a file in the users home directory. This file is used to hold information about fonts that isn't present in the per-directory cache files. It is automatically maintained by the fontconfig library. The default for this file is

“~/.fonts.cache-<version>”, where <version> is the font configuration file version number (currently 1).

<include ignore_missing="no">

This element contains the name of an additional configuration file. When the XML datatype is traversed by FcConfigParse, the contents of the file will also be incorporated into the configuration by passing the filename to FcConfigLoadAndParse. If ’ignore_missing’ is set to "yes" instead of the default "no", a missing file will elicit no warning message from the library.

<config>

This element provides a place to consolodate additional configuration information. <config> can contain <blank> and <rescan> elements in any order.

<blank>

Fonts often include "broken" glyphs which appear in the encoding but are drawn as blanks on the screen. Within the <blank> element, place each Unicode characters which is supposed to be blank in an <int> element. Characters outside of this set which are drawn as blank will be elided from the set of characters supported by the font. <b

<rescan>

The <rescan> element holds an <int> element which indicates the default interval between automatic checks for font configuration changes. Fontconfig will validate all of the configuration files and directories and automatically rebuild the internal datastructures when this interval passes.

<match target="pattern">

This element holds first a (possibly empty) list of <test> elements and then a (possibly empty) list of <edit> elements. Patterns which match all of the tests are subjected to all the edits. If ’target’ is set to "font" instead of the default "pattern", then this element applies to the font name resulting from a match rather than a font pattern to be matched.

<test qual="any" name="property" compare="eq">

This element contains a single value which is compared with the pattern property "property" (substitute any of the property names seen above). ’compare’ can be one of "eq", "not_eq", "less", "less_eq", "more", or "more_eq". ’qual’ may either be the default, "any", in which case the match succeeds if any value associated with the property matches the test value, or "all", in which case all of the values associated with the property must match the test value.

<edit name="property" mode="assign" binding="weak">

This element contains a list of expression elements (any of the value or operator elements). The expression elements are evaluated at run-time and modify the property "property". The modification depends on whether "property" was matched by one of the associated <test> elements, if so, the modification may affect the first matched value. Any values inserted into the property are given the indicated binding. ’mode’ is one of:

Mode	Operation with match	Operation without match
"assign"	Replace matching value	Replace all values
"assign_replace"	Replace all values	Replace all values
"prepend"	Insert before matching value	Insert at head of list
"prepend_first"	Insert at head of list	Insert at head of list
"append"	Append after matching value	Append at end of list
"append_last"	Append at end of list	Append at end of list

<int>
<double>
<string>
<bool>

These elements hold a single value of the indicated type. <bool> elements hold either true or false.

<matrix>

This element holds the four <double> elements of an affine transformation.

<name>

Holds a property name. Evaluates to the first value from the property of the font, not the pattern.

<const>

Holds the name of a constant; these are always integers and serve as symbolic names for common font values:

Constant	Property	CPP symbol
light	weight	FC_WEIGHT_LIGHT
medium	weight	FC_WEIGHT_MEDIUM
demibold	weight	FC_WEIGHT_DEMIBOLD
bold	weight	FC_WEIGHT_BOLD
black	weight	FC_WEIGHT_BLACK
roman	slant	FC_SLANT_ROMAN
italic	slant	FC_SLANT_ITALIC
oblique	slant	FC_SLANT_OBLIQUE
proportional	spacing	FC_PROPORTIONAL
mono	spacing	FC_MONO
charcell	spacing	FC_CHARCELL
unknown	rgba	FC_RGBA_UNKNOWN
rgb	rgba	FC_RGBA_RGB
bgr	rgba	FC_RGBA_BGR
vrgb	rgba	FC_RGBA_VRGB
vbgr	rgba	FC_RGBA_VBGR
none	rgba	FC_RGBA_NONE

<or>
<and>
<plus>
<minus>
<times>
<divide>

These elements perform the specified operation on a list of expression elements. <or> and <and> are boolean, not bitwise.

<eq>
<not_eq>
<less>
<less_eq>
<more>
<more_eq>

These elements compare two values, producing a boolean result.

<not>

Inverts the boolean sense of its one expression element

<if>

This element takes three expression elements; if the value of the first is true, it produces the value of the second, otherwise it produces the value of the third.

<alias>
Alias elements provide a shorthand notation for the set of common match operations needed to substitute one font family for another. They contain a <family> element followed by optional <prefer>, <accept> and <default> elements. Fonts matching the <family> element are edited to prepend the list of <prefer>ed families before the matching <family>, append the <accept>able familys after the matching <family> and append the <default> families to the end of the family list.

<family>
Holds a single font family name

<prefer>
<accept>
<default>
These hold a list of <family> elements to be used by the <alias> element.

EXAMPLE CONFIGURATION FILE

System configuration file

This is an example of a system-wide configuration file

```
<?xml version="1.0"?>
<!DOCTYPE fontconfig SYSTEM "fonts.dtd">
<!-- /etc/fonts/fonts.conf file to configure system font access -->
<fontconfig>
<!--
        Find fonts in these directories
-->
<dir>/usr/X11R6/lib/X11/fonts/truetype</dir>
<dir>/usr/X11R6/lib/X11/fonts/Type1</dir>

<!--
        Accept deprecated 'mono' alias, replacing it with 'monospace'
-->
<match target="pattern">
        <test qual="any" name="family"><string>mono</string></test>
        <edit name="family" mode="assign"><string>monospace</string></edit>
</match>

<!--
        Names not including any well known alias are given 'sans'
-->
<match target="pattern">
        <test qual="all" name="family" mode="not_eq">sans</test>
        <test qual="all" name="family" mode="not_eq">serif</test>
        <test qual="all" name="family" mode="not_eq">monospace</test>
        <edit name="family" mode="append_last"><string>sans</string></edit>
</match>

<!--
        Load per-user customization file, but don't complain
        if it doesn't exist
-->
<include ignore_missing="yes">~/.fonts.conf</include>

<!--
        Alias well known font names to available TrueType fonts.
        These substitute TrueType faces for similar Type1
```

```
        faces to improve screen appearance.
-->
<alias>
        <family>Times</family>
        <prefer><family>Times New Roman</family></prefer>
        <default><family>serif</family></default>
</alias>
<alias>
        <family>Helvetica</family>
        <prefer><family>Verdana</family></prefer>
        <default><family>sans</family></default>
</alias>
<alias>
        <family>Courier</family>
        <prefer><family>Courier New</family></prefer>
        <default><family>monospace</family></default>
</alias>

<!--
        Provide required aliases for standard names
        Do these after the users configuration file so that
        any aliases there are used preferentially
-->
<alias>
        <family>serif</family>
        <prefer><family>Times New Roman</family></prefer>
</alias>
<alias>
        <family>sans</family>
        <prefer><family>Verdana</family></prefer>
</alias>
<alias>
        <family>monospace</family>
        <prefer><family>Andale Mono</family></prefer>
</alias>
</fontconfig>
```

User configuration file

This is an example of a per-user configuration file that lives in ˜/.fonts.conf

```
<?xml version="1.0"?>
<!DOCTYPE fontconfig SYSTEM "fonts.dtd">
<!-- ˜/.fonts.conf for per-user font configuration -->
<fontconfig>

<!--
        Private font directory
-->
<dir>˜/misc/fonts</dir>

<!--
        use rgb sub-pixel ordering to improve glyph appearance on
        LCD screens.  Changes affecting rendering, but not matching
        should always use target="font".
-->
```

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

FILES

fonts.conf contains configuration information for the fontconfig library consisting of directories to look at for font information as well as instructions on editing program specified font patterns before attempting to match the available fonts. It is in xml format.

fonts.dtd is a DTD that describes the format of the configuration files.

local.conf is sourced by the default system-wide fonts.conf file. Note that the normal 'make install' procedure for XFree86 is to replace any existing fonts.conf file with the new version. Place any local customizations in local.conf which this file references.

~/.fonts.conf is the conventional location for per-user font configuration, although the actual location is specified in the global fonts.conf file.

~/.fonts.cache-* is the conventional repository of font information that isn't found in the per-directory caches. This file is automatically maintained by fontconfig.

AUTHOR

Keith Packard, member of the XFree86 Project, Inc.

NAME

fslsfonts − list fonts served by 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 Lists some attributes of the font on one line in addition to its name.

−ll Lists font properties in addition to **−l** output.

−lll Supported for compatibility with *xlsfonts*, but output is the same as for **−ll**.

−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. The default is 0, which 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

xfs(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.

AUTHOR

Dave Lemke, Network Computing Devices, Inc

NAME

fstobdf − generate BDF font from X font server

SYNOPSIS

fstobdf [**−server** *server*] **−fn** *fontname*

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.

ENVIRONMENT

FONTSERVER
: default server to use

SEE ALSO

xfs(1), bdftopcf(1), fslsfonts(1)

AUTHOR

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

Jim Fulton, MIT X Consortium

NAME

gccmakedep – create dependencies in makefiles using 'gcc -M'

SYNOPSIS

gccmakedep [**–s***separator*] [**–f***makefile*] [**–a**] [–– *options* ––] *sourcefile* ...

DESCRIPTION

The **gccmakedep** program calls 'gcc -M' to output *makefile* rules describing the dependencies of each *sourcefile*, so that **make**(1) knows which object files must be recompiled when a dependency has changed.

By default, **gccmakedep** 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 a line beginning with

```
# DO NOT DELETE
```

or one provided with the **–s** option, as a delimiter for the dependency output. If it finds it, it will delete everything following this up 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 makefile and place the output after that.

EXAMPLE

Normally, **gccmakedep** 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:
        gccmakedep –– $(CFLAGS) –– $(SRCS)
```

OPTIONS

The program will ignore any option that it does not understand, so you may use the same arguments that you would for **gcc**(1), including **–D** and **–U** options to define and undefine symbols and **–I** to set the include path.

–a Append the dependencies to the file instead of replacing existing dependencies.

–f*makefile*
Filename. This allows you to specify an alternate makefile in which **gccmakedep** can place its output. Specifying "–" as the file name (that is, **–f–**) sends the output to standard output instead of modifying an existing file.

–s*string*
Starting string delimiter. This option permits you to specify a different string for **gccmakedep** to look for in the makefile. The default is "# DO NOT DELETE".

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

SEE ALSO

gcc(1), **make**(1), **makedepend**(1).

AUTHOR

gccmakedep was written by the XFree86 Project based on code supplied by Hongjiu Lu.

Colin Watson wrote this manual page, originally for the Debian Project, based partly on the manual page for **makedepend**(1).

NAME

glxgears − GLX version of the infamous "gears" GL demo.

SYNOPSIS

glxgears [−info] [−display *displayname*]

DESCRIPTION

glxgears is a GLX demo that draws three rotating gears, and prints out framerate information to stdout. Command line options include:

−info Print out GL implementation information before running the demo.

−display *displayname*
Specify the display to query.

ENVIRONMENT

DISPLAY
To get the default host, display number, and screen.

SEE ALSO

glxinfo(1)

AUTHOR

Ported to straight GLX by Brian Paul.

NAME

glxinfo − display info about a GLX extension and OpenGL renderer.

SYNOPSIS

glxinfo [−t] [−v] [−b] [−display *displayname*]

DESCRIPTION

glxinfo lists information about the GLX extension, OpenGL capable visuals, and the OpenGL renderer on an X server. The GLX and renderer info includes the version and extension attributes. The visual info lists the GLX visual attributes available for each OpenGL capable visual (e.g. whether the visual is double buffered, the component sizes, Z-buffering depth, etc).

Command line options include:

−t By default the visual info is presented in a concise 80 character wide tabular format. The -t option directs glxinfo to produce a wider, more readable tabular format.

−v Directs glxinfo to generate a verbose format output style for the visual list similar to the info of xdpyinfo.

−b Print the ID of the "best" visual on screen 0.

−l Print interesting OpenGL limits.

−i Use indirect rendering connection only.

−display *displayname*
Specify the display to query.

ENVIRONMENT

DISPLAY
To get the default host, display number, and screen.

SEE ALSO

xdpyinfo(1)

AUTHOR

Brian Paul
Modifications for XFree86 added by Mark Paton

NAME

iceauth − ICE authority file utility

SYNOPSIS

iceauth [**−f** *authfile*] [**−vqib**] [*command arg ...*]

DESCRIPTION

The *iceauth* program is used to edit and display the authorization information used in connecting with ICE. This program is usually used to extract authorization records from one machine and merge them in on another (as is the case when using remote logins or granting access to other users). Commands (described below) may be entered interactively, on the *iceauth* command line, or in scripts.

AUTHOR

Ralph Mor, 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.

PROGRAM TREMINATION

Pressing "q" will close a window. If compiled with threads support, the program will stop only when all threads terminate. You can also close an animation window using the ICCCM *delete* message (depending on your window manager, you will have a decoration button or menu to send such message).

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(7)

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. Color allocation fails in TrueColor displays and option *-faces* does not work well.

COPYRIGHT

NAME

imake − C preprocessor interface to the make utility

SYNOPSIS

imake [**−D***define*] [**−I***dir*] [**−U***define*] [**−T***template*] [**−f** *filename*] [**−C** *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 as 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.

−U*define*
: This option is passed directly to *cpp*. It is typically used to unset variables when debugging *imake* configuration files.

−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*.

−C *filename*
: This option specifies the name of the .c file that is constructed in the current directory. The default is *Imakefile.c*.

−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 the name of a file containing 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)
```

Imake also replaces any occurrences of the word ‘‘XCOMM’’ with the character ‘‘#’’ to permit placing comments in the Makefile without causing ‘‘invalid directive’’ errors from the preprocessor.

Some complex *imake* macros require generated *make* variables local to each invocation of the macro, often because their value depends on parameters passed to the macro. Such variables can be created by using an *imake* variable of the form **XVARdef***n*, where *n* is a single digit. A unique *make* variable will be substituted. Later occurrences of the variable **XVARuse***n* will be replaced by the variable created by the corresponding **XVARdef***n*.

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
```

Imake.rules	rules
Project.tmpl	X-specific variables
*Lib.tmpl	shared library variables
Imakefile	
Library.tmpl	library rules
Server.tmpl	server rules
Threads.tmpl	multi-threaded 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

Imakefile.c	temporary input file for cpp
/tmp/Imf.XXXXXX	temporary Makefile for -s
/tmp/IIf.XXXXXX	temporary Imakefile if specified Imakefile uses # comments
__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 specifies a ‘‘−I’’ include argument to pass to the C preprocessor. E.g., ‘‘−I/usr/X11/config’’.

IMAKECPP

If defined, this should be a valid path to a preprocessor program. E.g., ‘‘/usr/local/cpp’’. By default, *imake* will use cc -E or __cpp__, depending on the OS specific configuration.

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

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)

NAME

lbxproxy - Low BandWidth X proxy

SYNOPSIS

lbxproxy [:<display>] [option]

DESCRIPTION

Applications that would like to take advantage of the Low Bandwidth extension to X (LBX) must make their connections to an lbxproxy. These applications need to know nothing about LBX, they simply connect to the lbxproxy as if were a regular server. The lbxproxy accepts client connections, multiplexes them over a single connection to the X server, and performs various optimizations on the X protocol to make it faster over low bandwidth and/or high latency connections.

With regard to authentication/authorization, lbxproxy simply passes along to the server the credentials presented by the client. Since X clients will connect to lbxproxy, it is important that the user's .Xauthority file contain entries with valid keys associated with the network ID of the proxy. lbxproxy does not get involved with how these entries are added to the .Xauthority file. The user is responsible for setting it up.

The lbxproxy program has various options, all of which are optional.

If **:<display>** is specified, the proxy will use the given display port when listening for connections. The display port is an offset from port 6000, identical to the way in which regular X display connections are specified. If no port is specified on the command line option, lbxproxy will default to port 63. If the port number that the proxy tries to listen on is in use, the proxy will attempt to use another port number. If the proxy is not using the Proxy Manager and the default port number cannot be used, the port number that is used will be written to stderr.

The other command line options that can be specified are:

−help Prints a brief help message about the command line options.

−display *dpy*
Specifies the address of the X server supporting the LBX extension. If this option is not specified, the display is obtained by the DISPLAY environment variable.

−motion *count*
A limited number of pointer motion events are allowed to be in flight between the server and the proxy at any given time. The maximum number of motion events that can be in flight is set with this option; the default is 8.

−maxservers *number*
The default behavior of lbxproxy is to manage a single server. However, lbxproxy can manage more than one server. The default maximum number of servers is 20. The number of servers can be overridden by setting the environment variable LBXPROXY_MAXSERVERS to the desired number. The order of precedence from highest to lowest: command line, environment variable, default number.

−[terminate|reset]
The default behavior of lbxproxy is to continue running as usual when it's last client exits. The **−terminate** option will cause lbxproxy to exit when the last client exits. The **−reset** option will cause lbxproxy to reset itself when the last client exits. Resetting causes lbxproxy to clean up it's state and reconnect to the server.

−reconnect
The default behavior of lbxproxy is to exit when its connection to the server is broken. The **−reconnect** option will cause lbxproxy to just reset instead (see **−reset** above) and attempt to reconnect to the server.

−I Causes all remaining arguments to be ignored.

−nolbx Disables all LBX optimizations.

−nocomp
Disables stream compression.

−nodelta
Disables delta request substitutions.

−notags Disables usage of tags.

−nogfx Disables reencoding of graphics requests (not including image related requests).

−noimage
Disables image compression.

−nosquish
Disables squishing of X events.

−nointernsc
Disables short circuiting of InternAtom requests.

−noatomsfile
Disables reading of the atoms control file. See the section on "Atom Control" for more details.

−atomsfile *file*
Overrides the default AtomControl file. See the section on "Atom Control" for more details.

−nowinattr
Disables GetWindowAttributes/GetGeometry grouping into one round trip.

−nograbcmap
Disables colormap grabbing.

−norgbfile
Disables color name to RGB resolution in proxy.

−rgbfile *path*
Specifies an alternate RGB database for color name to RGB resolution.

−tagcachesize
Set the size of the proxy's tag cache (in bytes).

−zlevel *level*
Set the Zlib compression level (used for stream compression).
default is 6
1 = worst compression, fastest
9 = best compression, slowest

−compstats
Report stream compression statistics every time the proxy resets or receives a SIGHUP signal.

−nozeropad
Don't zero out unused pad bytes in X requests, replies, and events.

−cheaterrors
Allows cheating on X protocol for the sake of improved performance. The X protocol guarantees that any replies, events or errors generated by a previous request will be sent before those of a later request. This puts substantial restrictions on when lbxproxy can short circuit a request. The -cheaterrors option allows lbxproxy to violate X protocol rules with respect to errors. Use at your own risk.

−cheatevents
The -cheatevents option allows lbxproxy to violate X protocol rules with respect to events as well as errors. Use at your own risk.

ATOM CONTROL

At startup, lbxproxy "pre-interns" a configurable list of atoms. This allows lbxproxy to intern a group of atoms in a single round trip and immediately store the results in its cache.

While running, lbxproxy uses heuristics to decide when to delay sending window property data to the server. The heuristics depend on the size of the data, the name of the property, and whether a window manager is running through the same lbxproxy.

Atom control is specified in the "AtomControl" file, set up during installation of lbxproxy, with command line overrides.

The file is a simple text file. There are three forms of lines: comments, length control, and name control. Lines starting with a '!' are treated as comments. A line of the form

> z *length*

specifies the minimum length in bytes before property data will be delayed. A line of the form

> *options atomname*

controls the given atom, where *options* is any combination of the following characters: 'i' means the atom should be pre-interned; and 'w' means data for properties with this name should be delayed only if a window manager is also running through the same lbxproxy.

BUGS

When the authorization protocol XDM-AUTHORIZATION-1 is used:

> A client must be on the same host as lbxproxy for the client to be authorized to connect to the server.
>
> If a client is not on the same host as lbxproxy, the client will not be authorized to connect to the server.

NAME

libxrx - RX Netscape Navigator Plug-in

DESCRIPTION

The **RX Plug-in** may be used with Netscape Navigator (3.0 or later) to interpret documents in the RX MIME type format and start remote applications.

The **RX Plug-in** reads an RX document, from which it gets the list of services the application wants to use. Based on this information, the **RX Plug-in** sets the various requested services, including creating authorization keys if your X server supports the SECURITY extension. It then passes the relevant data, such as the X display name, to the application through an HTTP GET request of the associated CGI script. The Web server then executes the CGI script to start the application. The client runs on the web server host connected to your X server. In addition when the RX document is used within the EMBED tag (a Netscape extension to HTML), the **RX Plug-in** uses the XC-APPGROUP extension, if it is supported by your X server, to cause the remote application to be embedded within the browser page from which it was launched.

INSTALLATION

To install the **RX Plug-in** so that Netscape Navigator can use it, find the file named libxrx.so.6.3 or libxrx.sl.6.3 (or similar, depending on your platform) in <ProjectRoot>/lib (e.g. /usr/X11R6.4/lib) and copy it to either /usr/local/lib/netscape/plugins or $HOME/.netscape/plugins. Do not install the symlinks libxrx.so or libxrx.sl; they would confuse Netscape.

If you have configured Netscape Navigator to use the RX helper program (**xrx**), you must reconfigure it. Generally you simply need to remove or comment out the line you may have previously added in your mailcap file to use the RX helper program. Otherwise the plug-in will not be enabled. (The usual comment character for mailcap is ‘‘#’’.)

If you are already running Netscape Navigator, you need to exit and restart it after copying the plug-in library so the new plug-in will be found. Once this is done you can check that Navigator has successfully loaded the plug-in by checking the ‘‘About Plug-ins’’ page from the Help menu. This should show something like:

```
                RX Plug-in

File name: /usr/local/lib/netscape/plugins/libxrx.sl.6.3

X Remote Activation Plug-in

Mime Type         Description                   Suffixes  Enabled
application/x-rx  X Remote Activation Plug-in   xrx       Yes
```

Once correctly configured, Netscape Navigator will activate the **RX Plug-in** whenever you retrieve any document of the MIME type *application/x-rx*.

RESOURCES

The **RX Plug-in** looks for resources associated with the widget **netscape.Navigator (**class **Netscape.TopLevelShell)** and understands the following resource names and classes:

xrxHasFirewallProxy (class **XrxHasFirewallProxy)**

Specifies whether an X server firewall proxy (see xfwp) is running and should be used. Default is ‘‘False.’’ The X firewall proxy uses the X Security Extension and this extension will only allow clients to connect to the X server if host-based authentication is turned on. See **xfwp(1)** for more information.

xrxInternalWebServers (class **XrxInternalWebServers)**

The web servers for which the X server firewall proxy should not be used (only relevant when **xrxHasFirewallProxy** is ‘‘True’’). Its value is a comma separated list of mask/value pairs to be used to filter internal web servers, based on their address. The mask part specifies which segments of the address are to be considered and the value part specifies what the result should match. For instance the following list:

255.255.255.0/198.112.45.0, 255.255.255.0/198.112.46.0

matches the address sets: 198.112.45.* and 198.112.46.*. More precisely, the test is (address & mask) == value.

xrxFastWebServers (class **XrxFastWebServers)**
The web servers for which LBX should not be used. The resource value is a list of address mask/value pairs, as previously described.

xrxTrustedWebServers (class **XrxTrustedWebServers)**
The web servers from which remote applications should be run as trusted clients. The default is to run remote applications as untrusted clients. The resource value is a list of address mask/value pairs, as previously described.

ENVIRONMENT

If the RX document requests X-UI-LBX service and the default X server does not advertise the LBX extension, the *RX Plug-in* will look for the environment variable ‘‘XREALDISPLAY’’ to get a second address for your X server and look for the LBX extension there. When running your browser through *lbxproxy* you will need to set XREALDISPLAY to the actual address of your server if you wish remote applications to be able to use LBX across the Internet.

If the RX document requests XPRINT service, *RX Plug-in* looks for the variable ‘‘XPRINTER’’ to get the printer name and X Print server address to use. If the server address is not specified as part of XPRINTER, *RX Plug-in* uses the first one specified through the variable ‘‘XPSERVERLIST’’ when it is set. When it is not *RX Plug-in* then tries to use the video server as the print server. If the printer name is not specified via XPRINTER, *RX Plug-in* looks for it in the variables ‘‘PDPRINTER’’, then ‘‘LPDEST’’, and finally ‘‘PRINTER’’,

Finally, if you are using a firewall proxy, *RX Plug-in* will look for ‘‘PROXY_MANAGER’’ to get the address of your proxy manager (see proxymngr). When not specified it will use ":6500" as the default.

KNOWN BUG

When an authorization key is created for a remote application to use the X Print service, the **RX Plug-in** has to create the key with an infinite timeout since nobody knows when the application will actually connect to the X Print server. It then revokes the key when its instance is destroyed (that is when you go to another page). However, if the Plug-in does not get destroyed properly, which happens when Netscape Navigator dies unexpectedly, the print authorization key will never get revoked.

SEE ALSO

xrx (1), xfwp (1), lbxproxy (1), proxymngr (1), The RX Document specification

AUTHORS

Arnaud Le Hors and Kaleb Keithley, 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(7), 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 [**−silent**] [**−ignorelinks**] [**−withrevinfo**] *fromdir* [*todir*]

DESCRIPTION

The *lndir* program 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 mounted from a remote machine. You can build in the shadow tree, and 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 scheme 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 all 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 BitKeeper, RCS, SCCS, CVS and CVS.adm directories are shadowed only if the **−withrevinfo** flag is specified.

If you add files, simply run *lndir* again. New files will be silently added. Old files will be checked that they have the correct link.

Deleting files is a more painful problem; the symlinks will just point into never never land.

If a file in *fromdir* is a symbolic link, *lndir* will make the same link in *todir* rather than making a link back to the (symbolic link) entry in *fromdir.* The **−ignorelinks** flag changes this behavior.

OPTIONS

−silent
Normally *lndir* outputs the name of each subdirectory as it descends into it. The **−silent** option suppresses these status messages.

−ignorelinks
Causes the program to not treat symbolic links in *fromdir* specially. The link created in *todir* will point back to the corresponding (symbolic link) file in *fromdir*. If the link is to a directory, this is almost certainly the wrong thing.

This option exists mostly to emulate the behavior the C version of *lndir* had in X11R6. Its use is not recommended.

−withrevinfo
Causes any BitKeeper, RCS, SCCS, CVS and CVS.adm subdirectories to be treated as any other directory, rather than ignored.

DIAGNOSTICS

The program displays the name of each subdirectory it enters, followed by a colon. The **−silent** option suppresses these messages.

A warning message is displayed if the symbolic link cannot be created. The usual problem is that a regular file of the same name already exists.

If the link already exists but doesn’t point to the correct file, the program prints the link name and the location where it does point.

NAME

luit − Locale and ISO 2022 support for Unicode terminals

SYNOPSIS

luit [*options*] [**−−**] [*program* [*args*]]

DESCRIPTION

Luit is a filter that can be run between an arbitrary application and a UTF-8 terminal emulator. It will convert application output from the locale's encoding into UTF-8, and convert terminal input from UTF-8 into the locale's encoding.

An application may also request switching to a different output encoding using ISO 2022 and ISO 6429 escape sequences. Use of this feature is discouraged: multilingual applications should be modified to directly generate UTF-8 instead.

Luit is usually invoked transparently by the terminal emulator. For information about running **luit** from the command line, see EXAMPLES below.

OPTIONS

−h Display some summary help and quit.

−list List the supported charsets and encodings, then quit.

−v Be verbose.

−c Function as a simple converter from standard input to standard output.

−x Exit as soon as the child dies. This may cause **luit** to loose data at the end of the child's output.

−argv0 *name*
Set the child's name (as passed in argv[0]).

−encoding *encoding*
Set up **luit** to use *encoding* rather than the current locale's encoding.

+oss Disable interpretation of single shifts in application output.

+ols Disable interpretation of locking shifts in application output.

+osl Disable interpretation of character set selection sequences in application output.

+ot Disable interpretation of all sequences and pass all sequences in application output to the terminal unchanged. This may lead to interesting results.

−k7 Generate seven-bit characters for keyboard input.

+kss Disable generation of single-shifts for keyboard input.

+kssgr Use GL codes after a single shift for keyboard input. By default, GR codes are generated after a single shift when generating eight-bit keyboard input.

−kls Generate locking shifts (SO/SI) for keyboard input.

−gl *gn* Set the initial assignment of GL. The argument should be one of **g0**, **g1**, **g2** or **g3**. The default depends on the locale, but is usually **g0**.

−gr *gk* Set the initial assignment of GR. The default depends on the locale, and is usually **g2** except for EUC locales, where it is **g1**.

−g0 *charset*
Set the charset initially selected in G0. The default depends on the locale, but is usually **ASCII**.

−g1 *charset*
Set the charset initially selected in G1. The default depends on the locale.

−g2 *charset*
Set the charset initially selected in G2. The default depends on the locale.

−g3 *charset*
Set the charset initially selected in G3. The default depends on the locale.

−ilog *filename*
Log into *filename* all the bytes received from the child.

−olog *filename*
Log into *filename* all the bytes sent to the terminal emulator.

−− End of options.

EXAMPLES

The most typical use of **luit** is to adapt an instance of **XTerm** to the locale's encoding. Current versions of **XTerm** invoke **luit** automatically when it is needed. If you are using an older release of **XTerm**, or a different terminal emulator, you may invoke **luit** manually:

```
$ xterm −u8 −e luit
```

If you are running in a UTF-8 locale but need to access a remote machine that doesn't support UTF-8, **luit** can adapt the remote output to your terminal:

```
$ LC_ALL=fr_FR luit ssh legacy-machine
```

Luit is also useful with applications that hard-wire an encoding that is different from the one normally used on the system or want to use legacy escape sequences for multilingual output. In particular, versions of **Emacs** that do not speak UTF-8 well can use **luit** for multilingual output:

```
$ luit -encoding 'ISO 8859-1' emacs -nw
```

And then, in **Emacs**,

```
M-x set-terminal-coding-system RET iso-2022-8bit-ss2 RET
```

FILES

/usr/X11R6/lib/X11/fonts/encodings/encodings.dir
The system-wide encodings directory.

/usr/X11R6/lib/X11/locale/locale.alias
The file mapping locales to locale encodings.

SECURITY

On systems with SVR4 (“Unix-98”) ptys (Linux version 2.2 and later, SVR4), **luit** should be run as the invoking user.

On systems without SVR4 (“Unix-98”) ptys (notably BSD variants), running **luit** as an ordinary user will leave the tty world-writable; this is a security hole, and luit will generate a warning (but still accept to run). A possible solution is to make **luit** suid root; **luit** should drop privileges sufficiently early to make this safe. However, the startup code has not been exhaustively audited, and the author takes no responsibility for any resulting security issues.

Luit will refuse to run if it is installed setuid and cannot safely drop privileges.

BUGS

None of this complexity should be necessary. Stateless UTF-8 throughout the system is the way to go.

Charsets with a non-trivial intermediary byte are not yet supported.

Selecting alternate sets of control characters is not supported and will never be.

SEE ALSO

xterm(1), unicode(7), utf-8(7), charsets(7). *Character Code Structure and Extension Techniques (ISO 2022, ECMA-35). Control Functions for Coded Character Sets (ISO 6429, ECMA-48).*

AUTHOR

Luit was written by Juliusz Chroboczek <jch@xfree86.org> for the XFree86 project.

NAME

makeg – make a debuggable executable

SYNOPSIS

makeg [*make-options* . . .] [*targets* . . .]

DESCRIPTION

The *makeg* script runs *make,* passing it variable settings to create a debuggable target when used with a Makefile generated by *imake.* For example, it arranges for the C compiler to be called with the **–g** option.

ENVIRONMENT

MAKE The *make* program to use. Default ‘‘make’’.

GDB Set to a non-null value if using the *gdb* debugger on Solaris 2, which requires additional debugging options to be passed to the compiler.

SEE ALSO

make (1), imake (1)

NAME

makepsres − Build PostScript resource database file.

SYNOPSIS

makepsres [*options*] *directory ...*

DESCRIPTION

makepsres creates PostScript language resource database files. Resource database files can be used to specify the location of resources that are used by the font selection panel and other Adobe software. For a complete description of the resource location facilities in the Display PostScript system, see Appendix A and Appendix B of "Display PostScript Toolkit for X" in *Programming the Display PostScript System with X.*

makepsres creates a resource database file named *PSres.upr* that contains all the resources in all the *directory* path names specified on the command line.

If the list of directories contains **− , makepsres** reads from *stdin* and expects a list of directories separated by space, tab, or newline.

If the list of directories is empty, it is taken to be the current directory.

If all specified directories have a common initial prefix, **makepsres** extracts it as a directory prefix in the new resource database file.

makepsres normally acts recursively; it looks for resource files in subdirectories of any specified directory. This behavior can be overridden with the command line option **−nr.**

makepsres uses existing resource database files to assist in identifying files. By default, **makepsres** creates a new resource database file containing all of the following that apply:

Resource files found in the directories on the command line.

Resource files pointed to by the resource database files in the directories on the command line.

Resource entries found in the input resource database files. These entries are copied if the files they specify still exist and are located in directories not specified on the command line.

If you run **makepsres** in discard mode (with the **−d** option), it does not copy resource entries from the input resource database files. In that case, the output file consists only of entries from the directories on the command line. The input resource database files are only used to assist in identifying files.

If you run **makepsres** in keep mode (with the **−k** option), it includes in the output file all resource entries in the input resource database files, even entries for files that no longer exist or are located in directories specified on the command line.

makepsres uses various heuristics to identify files. A file that is of a private resource type or that does not conform to the standard format for a resource file must be specified in one of the following ways:

By running **makepsres** in interactive mode

By preloading the file into a resource database file used for input

By beginning the file with the following line:

%!PS-Adobe-3.0 Resource-<resource-type>

OPTIONS

−o *filename*

Writes the output to the specified filename. The construction "**−o −**" writes to stdout. If the **−o** option is not specified, **makepsres** creates a *PSres.upr* file in the current directory and writes the output to that file.

–f *filename*
: Uses information from the specified file to assist in resource typing. The file must be in resource database file format. Multiple **–f** options may be specified. The construction "**–f –**" uses *stdin* as an input file and may not be used if "**–**" is specified as a directory on the command line.

–dir *dirname*
: Specifies that *dirname* is a directory. Needed only in rare cases when *dirname* is the same as a command-line option such as **–nb.**

–d Specifies discard mode. The resulting output file consists solely of entries from the directories on the command line.

–e Marks the resulting *PSres.upr* file as exclusive. This option makes the resource location library run more quickly since it does not have to look for other resource database files. It becomes necessary, however, to run **makepsres** whenever new resources are added to the directory, even if the resources come with their own resource database file.

–i Specifies interactive mode. In interactive mode, you will be queried for the resource type of any encountered file that **makepsres** cannot identify. If **–i** is not specified, **makepsres** assumes an unidentifiable file is not a resource file.

–k Specifies keep mode.

–nb If the output file already exists, do not back it up.

–nr Specifies nonrecursive mode. **makepsres** normally acts recursively: it looks for resource files in subdirectories of any specified directory. If **–nr** is used, **makepsres** does not look in subdirectories for resource files.

–p Specifies no directory prefix. If **–p** is used, **makepsres** does not try to find a common directory prefix among the specified directories.

–q Quiet mode: ignores unidentifiable files instead of warning about them.

–s Specifies strict mode. If **–s** is used, **makepsres** terminates with an error if it encounters a file it cannot identify.

EXAMPLES

makepsres .
: Creates a resource database file that contains all the resources in the current directory.

makepsres –i –o local.upr /usr/local/lib/ps/fonts
: Runs **makepsres** in interactive mode and creates a resource database file named *local.upr,* which contains all the resources in the directory */usr/local/lib/ps/fonts.*

SEE ALSO

Programming the Display PostScript System with X (Addison-Wesley Publishing Company, Inc., 1993).

AUTHOR

Adobe Systems Incorporated

NOTES

PostScript and Display PostScript are trademarks of Adobe Systems Incorporated which may be registered in certain jurisdictions.

NAME

makestrs − makes string table C source and header(s)

SYNOPSIS

makestrs [-f source] [-abioptions ...]

DESCRIPTION

The *makestrs* command creates string table C source files and headers. If *-f source* is not specified *makestrs* will read from *stdin.* The C source file is always written to *stdout. makestrs* creates one or more C header files as specified in the source file. The following options may be specified: *-sparcabi, -intelabi, -functionabi, -arrayperabi,* and *-defaultabi.*

-sparcabi is used on SPARC platforms conforming to the SPARC Compliance Definition, i.e. SVR4/Solaris.

-intelabi is used on Intel platforms conforming to the System V Application Binary Interface, i.e. SVR4.

-earlyR6abi may be used in addition to *-intelabi* for situations where the vendor wishes to maintain binary compatibility between X11R6 public-patch 11 (and earlier) and X11R6 public-patch 12 (and later).

-functionabi generates a functional abi to the string table. This mechanism imposes a severe performance penalty and it's recommended that you not use it.

-arrayperabi results in a separate array for each string. This is the default behavior if makestrs was compiled with -DARRAYPERSTR (it almost never is).

-defaultabi forces the generation of the "normal" string table even if makestrs was compiled with -DARRAYPERSTR. Since makestrs is almost never compiled with -DARRAYPERSTR this is the default behavior if no abioptions are specified.

SYNTAX

The syntax for string-list file is (items in square brackets are optional):

```
#prefix <text>
#feature <text>
#externref <text>
#externdef [<text>]
[#ctempl <text>]

#file <filename>
#table <tablename>
[#htempl]
<text>
<text>
[#table <tablename>
<text>
<text>
 ...
#table <tablename>
 ...]
[#file <filename>
 ...]
```

In words you may have one or more #file directives. Each #file may have one or more #table directives.

The #prefix directive determines the string that makestr will prefix to each definition.

The #feature directive determines the string that makestr will use for the feature-test macro, e.g. X[TM]STRINGDEFINES.

The #externref directive determines the string that makestr will use for the extern clause, typically this will be "extern" but Motif wants it to be "externalref"

The #externdef directive determines the string that makestr will use for the declaration, typically this will be the null string (note that makestrs requires a trailing space in this case, i.e. "#externdef "), and Motif will use "externaldef(_xmstrings).

The #ctmpl directive determines the name of the file used as a template for the C source file that is generated

Each #file <filename> directive will result in a corresponding header file by that name containing the appropriate definitions as specified by command line options. A single C source file containing the declarations for the definitions in all the headers will be printed to stdout.

The #htmpl directive determines the name of the file used as a template for the C header file that is generated.

Each #table <tablename> directive will be processed in accordance with the ABI. On most platforms all tables will be catenated into a single table with the name of the first table for that file. To conform to the Intel ABI separate tables will be generated with the names indicated.

The template files specified by the #ctmpl and #htmpl directives are processed by copying line for line from the template file to the appropriate output file. The line containing the string *<<<STRING_TABLE_GOES_HERE>>>* is not copied to the output file. The appropriate data is then copied to the output file and then the remainder of the template file is copied to the output file.

BUGS

makestrs is not very forgiving of syntax errors. Sometimes you need a trailing space after # directives, other times they will mess you up. No warning messages are emitted.

SEE ALSO

SPARC Compliance Definition 2.2., SPARC International Inc., 535 Middlefield Road, Suite 210, Menlo Park, CA 94025

System V Application Binary Interface, Third Edition, ISBN 0-13-100439-5 UNIX Press, PTR Prentice Hall, 113 Sylvan Avenue, Englewood Cliffs, NJ 07632

System V Application Binary Interface, Third Edition, Intel386 Architecture Processor Supplement ISBN 0-13-104670-5 UNIX Press, PTR Prentice Hall, 113 Sylvan Avenue, Englewood Cliffs, NJ 07632

System V Application Binary Interface, Third Edition, SPARC Architecture Processor Supplement ISBN 0-13-104696-9 UNIX Press, PTR Prentice Hall, 113 Sylvan Avenue, Englewood Cliffs, NJ 07632

NAME

mergelib − merge one library into another

SYNOPSIS

mergelib *to-library from-library* [*object-filename-prefix*]

DESCRIPTION

The *mergelib* program merges objects from one library into another. The names of object files in *from-library* will be prefixed by *object-filename-prefix* ("_" by default) to avoid name clashes. The merged library will be left in *to-library*.

AUTHOR

Jim Fulton wrote the *mergelib* program for the X Consortium.

Colin Watson wrote this manual page, originally for the Debian Project.

NAME

makedepend – create dependencies in makefiles

SYNOPSIS

makedepend [**–D***name***=***def*] [**–D***name*] [**–I***includedir*] [**–Y***includedir*] [**–a**] [**–f***makefile*] [**–include** *file*] [**–o***objsuffix*] [**–p***objprefix*] [**–s***string*] [**–w***width*] [**–v**] [**–m**] [**––** *otheroptions* **––**] *sourcefile* ...

DESCRIPTION

The **makedepend** program reads each *sourcefile* in sequence and parses it like a C-preprocessor, processing all *#include, #define, #undef, #ifdef, #ifndef, #endif, #if, #elif* 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

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

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

–I*includedir*
: 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 the standard include directories (usually /usr/include and possibly a compiler-dependent directory).

–Y*includedir*
: Replace all of the standard include directories with the single specified include directory; you can omit the *includedir* to simply prevent searching the standard include directories.

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

–f*makefile*
Filename. This allows you to specify an alternate makefile in which **makedepend** can place its output. Specifying "–" as the file name (i.e., **–f–**) sends the output to standard output instead of modifying an existing file.

–include *file*
Process file as input, and include all the resulting output before processing the regular input file. This has the same affect as if the specified file is an include statement that appears before the very first line of the regular input file.

–o*objsuffix*
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.

–p*objprefix*
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.

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

–w*width*
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.

–v Verbose operation. This option causes **makedepend** to emit the list of files included by each input file.

–m Warn about multiple inclusion. This option causes **makedepend** to produce a warning if any input file includes another file more than once. In previous versions of **makedepend** this was the default behavior; the default has been changed to better match the behavior of the C compiler, which does not consider multiple inclusion to be an error. This option is provided for backward compatibility, and to aid in debugging problems related to multiple inclusion.

−− *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 encounters 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

makedepend parses, but does not currently evaluate, the SVR4 #predicate(token-list) preprocessor expression; such expressions are simply assumed to be true. This may cause the wrong *#include* directives to be evaluated.

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

mkcfm - create summaries of font metric files in CID font directories

SYNOPSIS

mkcfm [*CID-font-directory-name*]

DESCRIPTION

There is usually only one CID font directory on the X font path. It is usually called */usr/X11R6/lib/X11/fonts/CID*. If you do not specify an argument, *mkcfm* will try to go through the subdirectories of that directory, and create one summary of font metric files for each CIDFont (character descriptions) file and each CMap (Character Maps) file it finds. The summaries of font metric files are put in the existing CFM subdirectory. The CFM subdirectories are created when CID-keyed fonts are installed.

If you specify a CID font directory as an argument, *mkcfm* will try to go through the subdirectories of that directory, and create one summary of font metric files for each CIDFont file and each CMap file it finds. *mkcfm* will calculate the summaries of the font metric files stored in AFM subdirectories of the CID font directory.

Those summaries are needed by the rasterizer of CID-keyed fonts to speed up the response to X font calls. If those files do not exist, CID rasterizer will have to go through usually large font metric files, and calculate the summaries itself each time the font is called. You will notice a substantial wait on a call to a large CID-keyed font.

FILES

.afm files Each CID-keyed font file is supposed to have a font metric file (.afm file). *mkcfm* creates summary files (.cfm files) of those font metric files. *mkcfm* should be run whenever a change is made to the files stored in the subdirectories of the CID font directory. For example, it should be run when new CID fonts are installed.

.cfm files Summaries of font metric (.afm) files created by *mkcfm*.

SEE ALSO

The rasterizer for CID-keyed fonts in the directory *xc/lib/font/Type1*.

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 an index of X font files in a directory

SYNOPSIS

mkfontdir [**−n**] [**−x** *suffix*] [**−r**] [**−p** *prefix*] [**−e** *encoding-directory-name*] ... [**−−**] [*directory-name* ...]

DESCRIPTION

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 converted to lower case and used as font names, and, along with the name of the font file, are written out to the file "fonts.dir" in the directory. The X server and font server use "fonts.dir" to find font files.

The kinds of font files read by *mkfontdir* depend 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.

The first line of fonts.dir gives the number of fonts in the file. The remaining lines list the fonts themselves, one per line, in two fields. First is the name of the font file, followed by a space and the name of the font.

SCALABLE FONTS

Because scalable font files do not usually include the X font name, the file "fonts.scale" can be used to name the scalable fonts in the directory. The fonts listed in it are copied to fonts.dir by *mkfontdir*. "fonts.scale" has the same format as the "fonts.dir" file.

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 two white-space separated columns, the first containing aliases and the second containing font-name patterns. Lines beginning with "!" are comment lines and are ignored.

If neither the alias nor the value specifies the size fields of the font name, this is a scalable alias. A font name of any size that matches this alias will be mapped to the same size of the font that the alias resolves to.

When a font alias is used, the name it references is searched 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 it 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 its suffix) will be used as an alias for that font.

ENCODING FILES

The option **-e** can be used to specify a directory with encoding files. Every such directory is scanned for encoding files, the list of which is then written to an "encodings.dir" file in every font directory. The "encodings.dir" file is used by the server to find encoding information.

The "encodings.dir" file has the same format as "fonts.dir". It maps encoding names (strings of the form **CHARSET_REGISTRY−CHARSET_ENCODING**) to encoding file names.

OPTIONS

The following options are supported:

−e Specify a directory containing encoding files. The **−e** option may be specified multiple times, and all the specified directories will be read. The order of the entries is significant, as encodings found in earlier directories override those in later ones; encoding files in the same directory are discriminated by preferring compressed versions.

−n do not scan for fonts, do not write font directory files. This option is useful when generating encoding directories only.

−p Specify a prefix that is prepended to the encoding file path names when they are written to the "encodings.dir" file. The prefix is prepended as-is. If a '/' is required between the prefix and the path names, it must be supplied explicitly as part of the prefix.

−r Keep non-absolute encoding directories in their relative form when writing the "encodings.dir" file. The default is to convert relative encoding directories to absolute directories by prepending the current directory. The positioning of this options is significant, as this option only applies to subsequent **−e** options.

−x *suffix*
Ignore fonts files of type *suffix*.

−− End options.

FILES

fonts.dir List of fonts in the directory and the files they are stored in. Created by *mkfontdir*. Read by the X server and font server each time the font path is set (see xset(1)).

fonts.scale List of scalable fonts in the directory. Contents are copied to fonts.dir by *mkfontdir*.

fonts.alias List of font name aliases. Read by the X server and font server each time the font path is set (see xset(1)).

encodings.dir List of known encodings and the files they are stored in. Created by *mkfontdir*. Read by the X server and font server each time a font with an unknown charset is opened.

SEE ALSO

X(7), Xserver(1), xfs(1), xset(1)

NAME

mkfontscale − create an index of scalable font files for X

SYNOPSIS

mkfontscale [**−b**] [**−s**] [**−o** *filename*] [**−x** *suffix*] [**−a** *encoding*] ... [**−f** *fuzz*] [**−l**] [**−e** *directory*] [**−p** *prefix*] [**−r** *prefix*] [**−n** *prefix*] [**−−**] [*directory*] ...

DESCRIPTION

For each directory argument, *mkfontscale* reads all of the scalable font files in the directory. For every font file found, an X11 font name (XLFD) is generated, and is written together with the file name to a file **fonts.scale** in the directory.

The resulting **fonts.scale** file should be checked and possibly manually edited before being used as input for the **mkfontdir**(1) program.

OPTIONS

−b read bitmap fonts. By default, bitmap fonts are ignored.

−s ignore scalable fonts. By default, scalable fonts are read. If **−b** is set, this flag has the side effect of enabling the reading of **fonts.scale** files. **−o** *filename* send program output to *filename*; default is **fonts.scale** if bitmap fonts are not being read, and **fonts.dir** if they are. If *filename* is relative, it is created in the directory being processed. If it is the special value **−**, output is written to standard output.

−x *suffix*
exclude all files with the specified *suffix*

−a *encoding*
add *encoding* to the list of encodings searched for.

−f *fuzz* set the fraction of characters that may be missing in large encodings to *fuzz* percent. Defaults to 2%.

−l Write **fonts.dir** files suitable for implementations that cannot reencode legacy fonts (BDF and PCF). By default, it is assumed that the implementation can reencode Unicode-encoded legacy fonts.

-e specifies a directory with encoding files. Every such directory is scanned for encoding files, the list of which is then written to an "encodings.dir" file in every font directory.

-p Specifies a prefix that is prepended to the encoding file path names when they are written to the "encodings.dir" file. The prefix is prepended litterally: if a '/' is required between the prefix and the path names, it must be supplied explicitly as part of the prefix.

−r Keep non-absolute encoding directories in their relative form when writing the "encodings.dir" file. The default is to convert relative encoding directories to absolute directories by prepending the current directory. The positioning of this options is significant, as this option only applies to subsequent

−n do not scan for fonts, do not write font directory files. This option is useful when generating encoding directories only.

−− end of options.

SEE ALSO

X(7), Xserver(1), mkfontdir(1), ttmkfdir(1), xfs(1), xset(1)

NOTES

The format of the **fonts.scale**, **fonts.dir** and **encodings.dir** files is documented in the mkfontdir(1) manual page.

Mkfontscale will overwrite any **fonts.scale** file even if it has been hand-edited.

mkfontscale -b -s -l is equivalent to **mkfontdir**.

AUTHOR

Mkfontscale was written by Juliusz Chroboczek <jch@xfree86.org> for the XFree86 project. The functionality of this program was inspired by the **ttmkfdir** utility by Joerg Pommnitz.

NAME

mkhtmlindex − generate index files for HTML man pages

SYNOPSIS

mkhtmlindex *htmlmandir*

DESCRIPTION

The *mkhtmlindex* program generates index files for a directory of HTML-formatted manual pages. It searches for files whose names are of the form "name.1.html", and outputs index files "manindex1.html", "manindex.2.html", and so on, one for each manual volume. Empty index files will be removed. Names and descriptions are found by scanning the first *<H2>* section of each page.

OPTIONS

mkhtmlindex takes only one argument: the directory to process.

NOTES

This utility is currently rather specific to XFree86. In particular, the format of the index files it outputs is not configurable, nor is the HTML formatting it expects of manual pages.

AUTHOR

David Dawes wrote the *mkhtmlindex* program for XFree86.

Colin Watson wrote this manual page, originally for the Debian Project.

NAME

oclock − round X clock

SYNOPSIS

oclock [−option ...]

DESCRIPTION

Oclock simply displays the current time on an analog display.

OPTIONS

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

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

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

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

−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(7)*.

−display *display*
: specify the display to use; see *X(7)*.

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

−bw *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.

−shape causes the clock to use the Shape extension to create an oval window. This is the default unless the shapeWindow resource is set to false.

−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/X11R6/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(7), X Toolkit documentation

AUTHOR

Keith Packard, MIT X Consortium

NAME

proxymngr - proxy manager service

SYNOPSIS

proxymngr [**−config** *filename*] [**−timeout** *seconds*] [**−retries** *#*] [**−verbose**]

DESCRIPTION

The proxy manager (proxymngr) is responsible for resolving requests from xfindproxy (and other similar clients), starting new proxies when appropriate, and keeping track of all of the available proxy services. The proxy manager strives to reuse existing proxies whenever possible.

There are two types of proxies that the proxy manager deals with, *managed* and *unmanaged* proxies.

A *managed* proxy is a proxy that is started ‘‘on demand’’ by the proxy manager.

An *unmanaged* proxy, on the other hand, is started either at system boot time, or manually by a system administrator. The proxy manager is made aware of its existence, but no attempt is made by the proxy manager to start unmanaged proxies.

The command line options that can be specified to **proxymngr** are:

−config Used to override the default proxymngr config file. See below for more details about the config file.

−timeout
Sets the number of seconds between attempts made by the proxy manager to find an unmanaged proxy. The default is 10.

−retries Sets the maximum number of retries made by the proxy manager to find an an unmanaged proxy. The default is 3.

−verbose
Causes various debugging and tracing records to be displayed as requests are received and proxies are started.

Proxy Manager Config File

The proxy manager maintains a local configuration file describing the proxy services available. This configuration file is installed in */usr/X11R6/lib/X11/proxymngr/pmconfig* during the installation of proxymngr. The location of the configuration file can be overwritten using the **−config** command line option.

Aside from lines starting with an exclamation point for comments, each line of the configuration file describes either an unmanaged or managed proxy service.

For unmanaged proxies, the format is:

<service-name> unmanaged <proxy-address>

service-name is the name of the unmanaged proxy service, and must not contain any spaces, for example ‘‘XFWP’’. service-name is case insensitive.

proxy-address is the network address of the unmanaged proxy. The format of the address is specific to the service-name. For example, for the ‘‘XFWP’’ service, the proxy-address might be ‘‘firewall.x.org:100’’.

If there is more than one entry in the config file with the same unmanaged service-name, the proxy manager will try to use the proxies in the order presented in the config file.

For managed proxies, the format is:

<service-name> managed <command-to-start-proxy>

service-name is the name of the managed proxy service, and must not contain any spaces, for example ‘‘LBX’’. service-name is case insensitive.

command-to-start-proxy is the command executed by the proxy manager to start a new instance of the proxy. If command-to-start-proxy contains spaces, the complete command should be surrounded by single quotes. If desired, command-to-start-proxy can be used to start a proxy on a remote machine. The specifics of the remote execution method used to do this is not specified here.

EXAMPLE

Here is a sample configuration file:

```
! proxy manager config file
!
! Each line has the format:
!    <serviceName> managed <startCommand>
!        or
!    <serviceName> unmanaged <proxyAddress>
!
lbx managed /usr/X11R6/bin/lbxproxy
!
! substitute site-specific info
xfwp unmanaged firewall:4444
```

PROXY MANAGER DETAILS

When the proxy manager gets a request from xfindproxy (or another similar client), its course of action will depend on the service-name in question.

For a managed proxy service, the proxy manager will find out if any of the already running proxies for this service can handle a new request. If not, the proxy manager will attempt to start up a new instance of the proxy (using the command-to-start-proxy found in the config file). If that fails, an error will be returned to the caller.

For an unmanaged proxy service, the proxy manager will look in the config file to find all unmanaged proxies for this service. If there is more than one entry in the config file with the same unmanaged service-name, the proxy manager will try to use the proxies in the order presented in the config file. If none of the unmanaged proxies can satisfy the request, the proxy manager will timeout for a configurable amount of time (specified by **−timeout** or default of 10) and reattempt to find an unmanaged proxy willing to satisfy the request. The number of retries can be specified by the **−retries** argument, or a default of 3 will be used. If the retries fail, the proxy manager has no choice but to return an error to the caller (since the proxy manager can not start unmanaged proxy services).

BUGS

proxy manager listen port should be configurable.

−timeout and **−retries** is not implemented in proxymngr.

proxymngr does not utilize the ‘‘options’’ and ‘‘host’’ fields in the proxy management protocol GetProxyAddr request.

SEE ALSO

xfindproxy (1), xfwp (1), Proxy Management Protocol spec V1.0

AUTHOR

Ralph Mor, X Consortium

NAME

pswrap – creates C procedures from segments of PostScript language code

SYNOPSIS

pswrap [**–apr**] [**–o** *outputCfile*] [**–h** *outputHfile*] [**–s** *maxstring*] *inputfile*

DESCRIPTION

pswrap reads input from *inputfile* and creates C-callable procedures, known as wraps, that send PostScript language code to the PostScript interpreter. *inputfile* contains segments of PostScript language code wrapped with a C-like procedure syntax.

Wraps are the most efficient way for an application to communicate with the PostScript interpreter. For complete documentation of **pswrap** and the language it accepts, see "pswrap Reference Manual" in *Programming the Display PostScript System with X.*

OPTIONS

inputfile

A file that contains one or more wrap definitions. **pswrap** transforms the definitions in *inputfile* into C procedures. If no input file is specified, the standard input (which can be redirected from a file or pipe) is used. The input file can include text other than wrap definitions. **pswrap** converts wrap definitions to C procedures and passes the other text through unchanged. Therefore, it is possible to intersperse C-language source code with wrap definitions in the input file.

Note: Although C code is allowed in a pswrap input file, it is not allowed within a wrap body. In particular, no CPP macros (for example, #define) are allowed inside a wrap.

–a

Generates ANSI C procedure prototypes for procedure definitions in *outputCfile* and, optionally, *outputHfile.* The **–a** option allows compilers that recognize the ANSI C standard to do more complete type checking of parameters. The **–a** option also causes **pswrap** to generate const declarations.

Note: ANSI C procedure prototype syntax is not recognized by most non-ANSI C compilers, including many compilers based on the Portable C Compiler. Use the **–a** option only in conjunction with a compiler that conforms to the ANSI C Standard.

–h *outputHFile*

Generates a header file that contains extern declarations for non-static wraps. This file can be used in #include statements in modules that use wraps. If the **–a** option is specified, the declarations in the header file are ANSI C procedure prototypes. If the **–h** option is omitted, a header file is not produced.

–o *outputCFile*

Specifies the file to which the generated wraps and passed-through text are written. If omitted, the standard output is used. If the **–a** option is also specified, the procedure definitions generated by **pswrap** are in ANSI C procedure prototype syntax.

–p

Specifies that strings passed by wraps are padded so that each data object begins on a long-word (4-byte) boundary. This option allows wraps to run on architectures that restrict data alignment to 4-byte boundaries and improves performance on some other architectures.

–r

Generates reentrant code for wraps shared by more than one process (as in shared libraries). Reentrant code can be called recursively or by more than one thread. The **–r** option causes **pswrap** to generate extra code, so use it only when necessary.

–s *maxstring*

Sets the maximum allowable length of a PostScript string object or hexadecimal string object in the wrap body input. A syntax error is reported if a string is not terminated with) or > within *maxstring* characters. *maxstring* cannot be set lower than 80; the default is 200.

SEE ALSO

Programming the Display PostScript System with X (Addison-Wesley Publishing Company, Inc., 1993).

AUTHOR

Adobe Systems Incorporated

NOTES

PostScript and Display PostScript are trademarks of Adobe Systems Incorporated which may be registered in certain jurisdictions.

NAME

resize − set TERMCAP and terminal settings to current xterm window size

SYNOPSIS

resize [**−u** | **−c**] [**−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 Sun console escape sequences will be used instead of the VT100-style *xterm* escape codes. If *rows* and *columns* are given, *resize* will ask the *xterm* to resize itself. However, the window manager may choose to disallow the change.

Note that the Sun console escape sequences are recognized by XFree86 *xterm* and by *dtterm*. The *resize* program may be installed as *sunsize*, which causes makes it assume the **−s** option.

The *rows* and *columns* arguments must appear last; though they are normally associated with the **−s** option, they are parsed separately.

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)

NAME

revpath − generate a relative path that can be used to undo a change-directory

SYNOPSIS

revpath *path*

DESCRIPTION

The *revpath* program prints out a relative path that is the ‘‘reverse’’ or ‘‘inverse’’ of *path*. Start with two directories *top* and *bottom*, with the latter below the former, and *path* is the location of *bottom* relative to *top*. The output of *revpath* is the location of *top* relative to *bottom*. The resulting path contains a trailing ‘/’ character when the result is non-trivial. If *path* is equivalent to ‘.’, the resulting output is empty. If *path* is invalid in some way (e.g., doesn’t represent the path to a subdirectory) the output is also empty and no error messages are ever generated.

DIAGNOSTICS

There are no diagnostics. Error conditions are silently ignored, and the exit status is always 0.

BUGS

It isn’t possible to reverse arbitrary relative paths. If any path element between the two end points of *path* is a symbolic link, the results will probably be incorrect.

NAME

rstart - a sample implementation of a Remote Start client

SYNOPSIS

rstart [−c *context*] [−g] [−l *username*] [−v] *hostname command args ...*

DESCRIPTION

Rstart is a simple implementation of a Remote Start client as defined in "A Flexible Remote Execution Protocol Based on **rsh**". It uses *rsh* as its underlying remote execution mechanism.

OPTIONS

−c *context*
: This option specifies the *context* in which the command is to be run. A *context* specifies a general environment the program is to be run in. The details of this environment are host-specific; the intent is that the client need not know how the environment must be configured. If omitted, the context defaults to **X**. This should be suitable for running X programs from the host's "usual" X installation.

−g
: Interprets *command* as a *generic command*, as discussed in the protocol document. This is intended to allow common applications to be invoked without knowing what they are called on the remote system. Currently, the only generic commands defined are **Terminal**, **LoadMonitor**, **ListContexts**, and **ListGenericCommands**.

−l *username*
: This option is passed to the underlying *rsh*; it requests that the command be run as the specified user.

−v
: This option requests that *rstart* be verbose in its operation. Without this option, *rstart* discards output from the remote's *rstart* helper, and directs the *rstart* helper to detach the program from the *rsh* connection used to start it. With this option, responses from the helper are displayed and the resulting program is not detached from the connection.

NOTES

This is a trivial implementation. Far more sophisticated implementations are possible and should be developed.

Error handling is nonexistent. Without **−v**, error reports from the remote are discarded silently. With **−v**, error reports are displayed.

The $DISPLAY environment variable is passed. If it starts with a colon, the local hostname is prepended. The local domain name should be appended to unqualified host names, but isn't.

The $SESSION_MANAGER environment variable should be passed, but isn't.

X11 authority information is passed for the current display.

ICE authority information should be passed, but isn't. It isn't completely clear how *rstart* should select what ICE authority information to pass.

Even without **−v**, the sample *rstart* helper will leave a shell waiting for the program to complete. This causes no real harm and consumes relatively few resources, but if it is undesirable it can be avoided by explicitly specifying the "exec" command to the shell, eg

```
rstart somehost exec xterm
```

This is obviously dependent on the command interpreter being used on the remote system; the example given will work for the Bourne and C shells.

SEE ALSO

rstartd(1), rsh(1), A Flexible Remote Execution Protocol Based on **rsh**

AUTHOR

Jordan Brown, Quarterdeck Office Systems

NAME

rstartd - a sample implementation of a Remote Start rsh helper

SYNOPSIS

rstartd

rstartd.real [−c *configfilename*]

DESCRIPTION

Rstartd is an implementation of a Remote Start "helper" as defined in "A Flexible Remote Execution Protocol Based on **rsh**".

This document describes the peculiarities of *rstartd* and how it is configured.

OPTIONS

−c *configfilename*

This option specifies the "global" configuration file that *rstartd* is to read. Normally, *rstartd* is a shell script that invokes *rstartd.real* with the **-c** switch, allowing local configuration of the location of the configuration file. If *rstartd.real* is started without the -c option, it reads */usr/X11R6/lib/X11/rstart/config*.

INSTALLATION

It is critical to successful interoperation of the Remote Start protocol that *rstartd* be installed in a directory which is in the "default" search path, so that default rsh requests and the ilk will be able to find it.

CONFIGURATION AND OPERATION

Rstartd is by design highly configurable. One would like things like configuration file locations to be fixed, so that users and administrators can find them without searching, but reality is that no two vendors will agree on where things should go, and nobody thinks the original location is "right". Thus, *rstartd* allows one to relocate **all** of its files and directories.

Rstartd has a hierarchy of configuration files which are executed in order when a request is made. They are:

global config
per-user ("local") config
global per-context config
per-user ("local") per-context config
config from request

As you might guess from the presence of "config from request", all of the config files are in the format of an *rstart* request. *Rstartd* defines a few additional keywords with the INTERNAL- prefix for specifying its configuration.

Rstartd starts by reading and executing the global config file. This file will normally specify the locations of the other configuration files and any systemwide defaults.

Rstartd will then read the user's local config file, default name $HOME/.rstart.

Rstartd will then start interpreting the request.

Presumably one of the first lines in the request will be a CONTEXT line. The context name is converted to lower case.

Rstartd will read the global config file for that context, default name /usr/X11R6/lib/X11/rstart/contexts/<name>, if any.

It will then read the user's config file for that context, default name $HOME/.rstart.contexts/<name>, if any.

(If neither of these exists, *rstartd* aborts with a Failure message.)

Rstartd will finish interpreting the request, and execute the program specified.

This allows the system administrator and the user a large degree of control over the operation of *rstartd*. The administrator has final say, because the global config file doesn't need to specify a per-user config file. If it does, however, the user can override anything from the global file, and can even completely replace the global context config files.

The config files have a somewhat more flexible format than requests do; they are allowed to contain blank lines and lines beginning with "#" are comments and ignored. (#s in the middle of lines are data, not comment markers.)

Any commands run are provided a few useful pieces of information in environment variables. The exact names are configurable, but the supplied defaults are:

$RSTART_CONTEXT	the name of the context
$RSTART_GLOBAL_CONTEXTS	the global contexts directory
$RSTART_LOCAL_CONTEXTS	the local contexts directory
$RSTART_GLOBAL_COMMANDS	the global generic commands directory
$RSTART_LOCAL_COMMANDS	the local generic commands directory

$RSTART_{GLOBAL,LOCAL}_CONTEXTS should contain one special file, @List, which contains a list of the contexts in that directory in the format specified for ListContexts. The supplied version of ListContexts will cat both the global and local copies of @List.

Generic commands are searched for in several places: (defaults)

per-user per-context directory ($HOME/.rstart.commands/<context>)
global per-context directory (/usr/X11R6/lib/X11/rstart/commands/<context>)
per-user all-contexts directory ($HOME/.rstart.commands)
global all-contexts directory (/usr/X11R6/lib/X11/rstart/commands)

(Yes, this means you can't have an all-contexts generic command with the same name as a context. It didn't seem like a big deal.)

Each of these directories should have a file called @List that gives the names and descriptions of the commands in that directory in the format specified for ListGenericCommands.

CONFIGURATION KEYWORDS

There are several "special" *rstart* keywords defined for *rstartd* configuration. Unless otherwise specified, there are no defaults; related features are disabled in this case.

INTERNAL-REGISTRIES name ...
Gives a space-separated list of "MISC" registries that this system understands. (Registries other than this are accepted but generate a Warning.)

INTERNAL-LOCAL-DEFAULT relative_filename
Gives the name ($HOME relative) of the per-user config file.

INTERNAL-GLOBAL-CONTEXTS absolute_directory_name
Gives the name of the system-wide contexts directory.

INTERNAL-LOCAL-CONTEXTS relative_directory_name
Gives the name ($HOME relative) of the per-user contexts directory.

INTERNAL-GLOBAL-COMMANDS absolute_directory_name
Gives the name of the system-wide generic commands directory.

INTERNAL-LOCAL-COMMANDS relative_directory_name
Gives the name ($HOME relative) of the per-user generic commands directory.

INTERNAL-VARIABLE-PREFIX prefix
Gives the prefix for the configuration environment variables *rstartd* passes to its kids.

INTERNAL-AUTH-PROGRAM authscheme program argv[0] argv[1] ...
Specifies the program to run to set up authentication for the specified authentication scheme. "program argv[0] ..." gives the program to run and its arguments, in the same form as the EXEC keyword.

INTERNAL-AUTH-INPUT authscheme

Specifies the data to be given to the authorization program as its standard input. Each argument is passed as a single line. $n, where n is a number, is replaced by the n'th argument to the "AUTH authscheme arg1 arg2 ..." line.

INTERNAL-PRINT arbitrary text

Prints its arguments as a Debug message. Mostly for *rstartd* debugging, but could be used to debug config files.

NOTES

When using the C shell, or any other shell which runs a script every time the shell is started, the script may get run several times. In the worst case, the script may get run **three** times:

By rsh, to run *rstartd*
By *rstartd*, to run the specified command
By the command, eg *xterm*

rstartd currently limits lines, both from config files and requests, to BUFSIZ bytes.

DETACH is implemented by redirecting file descriptors 0,1, and 2 to /dev/null and forking before executing the program.

CMD is implemented by invoking $SHELL (default /bin/sh) with "-c" and the specified command as arguments.

POSIX-UMASK is implemented in the obvious way.

The authorization programs are run in the same context as the target program - same environment variables, path, etc. Long term this might be a problem.

In the X context, GENERIC-CMD Terminal runs xterm. In the OpenWindows context, GENERIC-CMD Terminal runs cmdtool.

In the X context, GENERIC-CMD LoadMonitor runs xload. In the OpenWindows context, GENERIC-CMD LoadMonitor runs perfmeter.

GENERIC-CMD ListContexts lists the contents of @List in both the system-wide and per-user contexts directories. It is available in all contexts.

GENERIC-CMD ListGenericCommands lists the contents of @List in the system-wide and per-user commands directories, including the per-context subdirectories for the current context. It is available in all contexts.

CONTEXT None is not implemented.

CONTEXT Default is really dull.

For installation ease, the "contexts" directory in the distribution contains a file "@Aliases" which lists a context name and aliases for that context. This file is used to make symlinks in the contexts and commands directories.

All **MISC** values are passed unmodified as environment variables.

One can mistreat *rstartd* in any number of ways, resulting in anything from stupid behavior to core dumps. Other than by explicitly running programs I don't think it can write or delete any files, but there's no guarantee of that. The important thing is that (a) it probably won't do anything REALLY stupid and (b) it runs with the user's permissions, so it can't do anything catastrophic.

@List files need not be complete; contexts or commands which are dull or which need not or should not be advertised need not be listed. In particular, per-user @List files should not list things which are in the system-wide @List files. In the future, perhaps ListContexts and ListGenericCommands will automatically suppress lines from the system-wide files when there are per-user replacements for those lines.

Error handling is OK to weak. In particular, no attempt is made to properly report errors on the exec itself. (Perversely, exec errors could be reliably reported when detaching, but not when passing the stdin/out socket to the app.)

If compiled with -DODT1_DISPLAY_HACK, *rstartd* will work around a bug in SCO ODT version 1. (1.1?) (The bug is that the X clients are all compiled with a bad library that doesn't know how to look host names up using DNS. The fix is to look up a host name in $DISPLAY and substitute an IP address.) This is a trivial example of an incompatibility that *rstart* can hide.

SEE ALSO

rstart(1), rsh(1), A Flexible Remote Execution Protocol Based on **rsh**

AUTHOR

Jordan Brown, Quarterdeck Office Systems

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 and Linux also have 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/X11R6/lib/xdm/Xservers $USER
```

and in Xreset:

```
sessreg -d -l $DISPLAY -x /usr/X11R6/lib/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)

AUTHOR

Keith Packard, MIT X Consortium

NAME

setxkbmap − set the keyboard using the X Keyboard Extension

SYNOPSIS

setxkbmap [*args* **] [** *layout* **[** *variant* **[** *option ...* **]]]**

DESCRIPTION

The **setxkbmap** command maps the keyboard to use the layout determined by the options specified on the command line.

An XKB keymap is constructed from a number of components which are compiled only as needed. The source for all of the components can be found in */usr/X11R6/lib/X11/xkb*.

OPTIONS

−help Prints a message describing the valid input to *setxkbmap*.

−compat *name*
Specifies the name of the compatibility map component used to construct a keyboard layout.

−config *file*
Specifies the name of an XKB configuration file which describes the keyboard to be used.

−display *display*
Specifies the display to be updated with the new keyboard layout.

−geometry *name*
Specifies the name of the geometry component used to construct a keyboard layout.

−keymap *name*
Specifies the name of the keymap description used to construct a keyboard layout.

−layout *name*
Specifies the name of the layout used to determine the components which make up the keyboard description. Only one layout may be specified on the command line.

−model *name*
Specifies the name of the keyboard model used to determine the components which make up the keyboard description. Only one model may be specified on the command line.

−option *name*
Specifies the name of an option to determine the components which make up the keyboard description; multiple options may be specified, one per *-option* flag. Note that **setxkbmap** summarize options specified in the command line with options was set before (saved in root window properties). If you want only specified options will be set use the *-option* flag with an empty argument first.

−print With this option the **setxkbmap** just prints component names in a format acceptable by an **xkbcomp** (an XKB keymap compiler) and exits. The option can be used for tests instead of a verbose option and in case when one need to run both the **setxkbmap** and the **xkbcomp** in chain (see below).

−rules *file*
Specifies the name of the rules file used to resolve the request layout and model to a set of component names.

−symbols *name*
Specifies the name of the symbols component used to construct a keyboard layout.

−synch Force synchronization for X requests.

−types *name*
Specifies the name of the types component used to construct a keyboard layout.

−variant *name*
Specifies which variant of the keyboard layout should be used to determine the components which make up the keyboard description. Only one variant may be specified on the command line.

USING WITH xkbcomp

If you have an Xserver and a client shell running on different computers and XKB configuration files sets on those machines are different you can get problems specifying a keyboard map by model, layout, options names. The thing is the **setxkbcomp** converts these names to names of XKB configuration files according to files that are on the client side computer. Then it sends the file names to the server where the **xkbcomp** has to compose a complete keyboard map using files which the server has. Thus if the sets of files differ significantly the names that the **setxkbmap** generates can be unacceptable on the server side. You can solve this problem running the **xkbcomp** on the client side too. With the *-print* option **setxkbmap** just prints the files names in an appropriate format to its stdout and this output can be piped directly to the **xkbcomp** input. For example, a command

setxkbmap us -print | xkbcomp - $DISPLAY

makes both step on the same (client) machine and loads a keyboard map into the server.

FILES

/usr/X11R6/lib/X11/xkb

NAME

showfont − font dumper for X font server

SYNOPSIS

showfont [−*options* . . .] **−fn** *pattern*

DESCRIPTION

Showfont displays data about a font that matches the given *pattern*. The information shown includes font information, font properties, character metrics, and character bitmaps.

The wildcard character "*" can be used to match any sequence of characters (including none) in the font name, and "?" can be used to match any single character. The "*" and "?" characters must be quoted to prevent them from being expanded by the shell. If no pattern is given, "*" is assumed.

OPTIONS

−server *host***:***port*
The X font server to contact.

−fn *name*
The font to display.

−lsb The bit order of the font should be requested as LSBFirst (least significant bit first).

−msb The bit order of the font should be requested as MSBFirst (most significant bit first).

−LSB The byte order of the font should be requested as LSBFirst (least significant byte first).

−MSB The byte order of the font should be requested as MSBFirst (most significant byte first).

−ext[**ents_only**]
Only the character extents should be displayed, but not the bitmaps.

−start *char*
The start of the range of the characters to display (*char* is a number).

−end *char*
The end of the range of the characters to display (*char* is a number).

−unit *n* The scanline unit of the font (8, 16, 32, or 64).

−pad *n* The scanpad unit of the font (8, 16, 32, or 64).

−b[**itmap_pad**] *n*
The bitmap padding unit of the font (0, 1, or 2, where 0 is ImageRectMin, 1 is ImageRectMaxWidth and 2 is ImageRectMax).

−noprops
Do not show the font properties.

SEE ALSO

xfs(1), fslsfonts(1), xlsfonts(1)

ENVIRONMENT

FONTSERVER
the default X font server to contact.

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/X11R6/lib/X11/rgb
default database.

NAME

smproxy − Session Manager Proxy

SYNOPSIS

smproxy [-clientId id] [-restore saveFile]

OPTIONS

−clientId *id*

Specifies the session ID used by *smproxy* in the previous session.

−restore *saveFile*

Specifies the file used by *smproxy* to save state in the previous session.

DESCRIPTION

smproxy allows X applications that do not support X11R6 session management to participate in an X11R6 session.

In order for *smproxy* to act as a proxy for an X application, one of the following must be true:

- The application maps a top level window containing the **WM_CLIENT_LEADER** property. This property provides a pointer to the client leader window which contains the **WM_CLASS**, **WM_NAME**, **WM_COMMAND**, and **WM_CLIENT_MACHINE** properties.

or ...

- The application maps a top level window which does not contain the **WM_CLIENT_LEADER** property. However, this top level window contains the **WM_CLASS**, **WM_NAME**, **WM_COMMAND**, and **WM_CLIENT_MACHINE** properties.

An application that support the **WM_SAVE_YOURSELF** protocol will receive a **WM_SAVE_YOURSELF** client message each time the session manager issues a checkpoint or shutdown. This allows the application to save state. If an application does not support the **WM_SAVE_YOURSELF** protocol, then the proxy will provide enough information to the session manager to restart the application (using **WM_COMMAND**), but no state will be restored.

SEE ALSO

xsm(1)

AUTHOR

Ralph Mor, X Consortium

NAME

startx − initialize an X session

SYNOPSIS

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

DESCRIPTION

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 often run with no arguments.

Arguments immediately following the *startx* command are used to start a client in the same manner as *xinit*(1). The special argument '--' marks the end of client arguments and the beginning of server options. It may be convenient to specify server options with startx to change on a per-session basis the default color depth, the server's notion of the number of dots-per-inch the display device presents, or take advantage of a different server layout, as permitted by the *XFree86*(1) server and specified in the *XF86Config*(5) file. Some examples of specifying server arguments follow; consult the manual page for your X server to determine which arguments are legal.

startx -- -depth 16

startx -- -dpi 100

startx -- -layout Multihead

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 and revert to the *xinit*(1) 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 and revert to the *xinit*(1) behavior. Users rarely need to provide a *.xserverrc* file. See the *xinit*(1) manual page for more details on the arguments.

The system-wide *xinitrc* and *xserverrc* files are found in the */usr/X11R6/lib/X11/xinit* directory.

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.

XAUTHORITY
: This variable, if not already defined, gets set to *$(HOME)/.Xauthority*. This is to prevent the X server, if not given the −*auth* argument, from automatically setting up insecure host-based authentication for the local

host. See the *Xserver*(1) and *Xsecurity*(7) manual pages for more information on X client/server authentication.

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/X11R6/lib/X11/xinit/xinitrc
: Client to run if the user has no *.xinitrc* file.

/usr/X11R6/lib/X11/xinit/xserverrc
: Server to run if the user has no *.xserverrc* file.

SEE ALSO

xinit(1), *Xserver*(1), *XFree86*(1)

NAME

sxpm − Show an XPM (X PixMap) file and/or convert XPM 1 or 2 files to XPM 3.

SYNOPSIS

sxpm [**-d** displayname] [**-g** geometry] [**-hints**] [**-icon** filename] [**-plaid** | filename | -] [**-o** filename | **-o** -] [**-pcmap**] [**-closecolors**] [**-nod**] [**-nom**] [**-mono | -grey4 | -grey | -color**] [**-sc** symbol color] [**-sp** symbol pixel] [**-cp** color pixel] [**-rgb** filename] [**-v**]

DESCRIPTION

The *sxpm* program can be used to view any XPM (version 1, 2, or 3) file and/or to convert a file from XPM1 or XPM2 to XPM version 3. If *sxpm* is run with any dummy option specified, the usage is displayed. If no geometry is specified, the show window will have the size of the read pixmap. Pressing the key Q in the window will quit the program.

OPTIONS

−d *display*
Specifies the display to connect to.

−g *geom* Window geometry (default is pixmap's size).

−hints Set ResizeInc for window.

−icon *filename*
Set icon to pixmap created from the file *filename*.

−plaid Show the plaid pixmap which is stored as data**.**

filename Read from the file *filename* and from standard input if *filename* is '-'. If no input is specified sxpm reads from standard input.

−o *filename*
Write to the file *filename* (overwrite if it already exists) and to standard output if *filename* is '-'.

−mono Use the colors specified for a monochrome visual.

−grey4 Use the colors specified for a 4 color greyscale visual.

−grey Use the colors specified for a greyscale visual.

−color Use the colors specified for a color visual.

−pcmap Use a private colormap.

−closecolors
Try to use "close colors" before reverting to other visuals.

−nod Do not display the pixmap in a window. (Useful when using as converter)

−nom Do not use the clipmask if there is any.

−sc *symbol colorname*
Override default color to *symbol* to *colorname*.

−sp *symbol pixelvalue*
Override default color to *symbol* to *pixelvalue*.

−cp *colorname pixelvalue*
Override default color to *colorname* to *pixelvalue*.

−rgb *filename*
Search color names in the file *filename* and write them out instead of the rgb values.

−v Verbose - to print out extensions (stderr).

KNOWN BUGS

Some window managers may not accept a pixmap which is not a bitmap as icon because this does not respect ICCCM, many of the well known ones will accept it though.

AUTHOR

Arnaud Le Hors (lehors@sophia.inria.fr)
Bull Research France

NAME

testplugin - a Netscape Plug-in test bed utility

SYNOPSIS

testplugin src=*url* [width=*width*] [height=*height*] [*name*=*value*] ...

DESCRIPTION

This program is designed to provide a means for testing Netscape Navigator UNIX plug-ins. It exercises the plug-in in a way close (I hope) to how Navigator does, and because it is a standalone program it allows you to run it through debugger and to use various program analysis tools.

The line-mode browser **www**, must be in your **PATH** to be able to use testplugin successfully.

ARGUMENTS

src=*url* This argument specifies the source document to use. It should be of the MIME type your plug-in is expecting since unlike within Netscape, no type checking is done and the document is simply streamed to the plug-in.

[width=*width***] [height=***height***]**
These options allow to specify the plug-in window size.

[*name***=***value***]...**
In addition to the arguments described above, any other argument can be specified and will be passed uninterpreted to the plug-in (through the NPP_New method).

CURRENT LIMITATIONS

I've not implemented all of the "Netscape Methods", but only the ones I've needed so far. Also the "Plug-in Methods" are not called in every possible manner so it does not provide a 100% testing coverage.

SEE ALSO

www(1), Netscape Navigator Documentation

AUTHOR

Arnaud Le Hors, X Consortium

NAME

texteroids − test your mousing skills on spinning text

SYNOPSIS

texteroids [**−display** *name*][**−fn** *font*][**−size** *size*][*text_string*]

DESCRIPTION

texteroids spins the specified text string in a window. If you click on the text with the mouse, the string splits up into individual letters, each of which you may then click on.

OPTIONS

−display *name*
specifies the display on which to open a connection to the Display PostScript System. If no display is specified, the DISPLAY environment variable is used.

−fn *font*
specifies the name of the PostScript language font software to use. The default is Times-Italic.

−size *size*
specifies the size, in points, of the text. The default is 36.

−debug
specifies debugging mode. In debugging mode, all PostScript code sent to the server is printed out.

text_string
specifies the text to display. If the text has spaces it must be enclosed in quotation marks. The default text string is "Adobe".

AUTHOR

Adobe Systems Incorporated

NOTES

PostScript and Display PostScript are trademarks of Adobe Systems Incorporated which may be registered in certain jurisdictions.

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/X11R6/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 (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 **:delete** (both refer to the X logo), **:dot** or **:iconify** (both refer to the dot), **:resize** (the nested squares used by the resize button), **:menu** (a page with lines), 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**, **Grayscale** 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**, **Grayscale** 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**, **Grayscale** 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**, **MenuShadowColor**, **MenuBorderColor**, **PointerForeground**, and **PointerBackground**. 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 to be moved only 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/X11R6/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.

Grayscale { *colors* }
This variable specifies a list of color assignments that should be made if the screen has a GrayScale default visual. See the description of **Colors**.

IconBackground *string* [{ *win-list* }]
This variable specifies the background color of icons, and may only be specified inside of a **Color**, **Grayscale** 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**, **Grayscale** 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**, **Grayscale** 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 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**, **Grayscale** 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**, **Grayscale** 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**, **Grayscale** 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

IconManagerGeometry. 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 width is 32767 - screen width. The default height is 32767 - screen height.

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".

MenuBorderColor *string*
This variable specifies the color of the menu border and can only be specified inside of a **Color**, **Grayscale** or **Monochrome** list. The default is "black".

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

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**, **Grayscale** 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**, **Grayscale** 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**, **Grayscale** 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 a 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.

Priority *priority*
This variable sets *twm*'s priority. *priority* should be an unquoted, signed number (e.g. 999). This variable has an effect only if the server supports the SYNC extension.

RandomPlacement
This variable indicates that windows with no specified geometry 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**, **Grayscale** 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 be 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**, **Grayscale** 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 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. If the server supports the SYNC extension, the priority of the client owning the window is also displayed. 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.priority *string*
This function sets the priority of the client owning the selected window to the numeric value of the argument *string*, which should be a signed integer in double quotes (e.g. "999"). This function has an effect only if the server supports the SYNC extension.

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.startwm *string*
This function kills *twm* and starts another window manager, as specified by *string*.

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/X11R6/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(7), Xserver(1), xdm(1), xrdb(1)

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

ucs2any − generate BDF fonts containing subsets of ISO 10646-1 codepoints

SYNOPSIS

ucs2any [**+d | -d**] *source-name* { *mapping-file registry-encoding* } . . .

DESCRIPTION

ucs2any allows one to generate from an ISO 10646-1 encoded BDF font other BDF fonts in any possible encoding. This way, one can derive from a single ISO 10646-1 master font a whole set of 8-bit fonts in all ISO 8859 and various other encodings.

OPTIONS

+d puts DEC VT100 graphics characters in the C0 range (default for upright, character-cell fonts).

−d omits DEC VT100 graphics characters from the C0 range (default for all font types except upright, character-cell fonts).

OPERANDS

source-name
is the name of an ISO 10646-1 encoded BDF file.

mapping-file
is the name of a character set table like those at **<ftp://ftp.unicode.org/Public/MAPPINGS/>**. These files can also typically be found installed in the */usr/X11R6/lib/X11/fonts/util/* directory.

registry-encoding
are the CHARSET_REGISTRY and CHARSET_ENCODING field values for the font name (XLFD) of the target font, separated by a hyphen.

Any number of *mapping-file* and *registry-encoding* operand pairs may be specified.

EXAMPLE

The command

```
ucs2any 6x13.bdf 8859-1.TXT iso8859-1 8859-2.TXT iso8859-2
```

will generate the files *6x13-iso8859-1.bdf* and *6x13-iso8859-2.bdf* .

FUTURE DIRECTIONS

Hopefully a future XFree86 release will have a facility similar to **ucs2any** built into the server, and reencode ISO 10646-1 on the fly, because storing the same fonts in many different encodings is clearly a waste of storage capacity.

SEE ALSO

bdftruncate(1)

AUTHOR

ucs2any was written by Markus Kuhn.

Branden Robinson wrote this manual page, originally for the Debian Project.

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(7), 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 performance. *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, Chris Kent, Mary Walker, and Todd Newman, who wanted to assess performance differences between various servers. 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-or-pixel*
: Specifies the foreground color or pixel value to use.

–bg *color-or-pixel*
: Specifies the background color or pixel value to use.

–clips *default*
: Default number of clip windows.

–ddbg *color-or-pixel*
: Specifies the color or pixel value to use for drawing the odd segments of a DoubleDashed line or arc. This will default to the bg color.

–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).

−vclass <vclass>
Use a visual with of class <vclass>. <vclass> can be StaticGray, GrayScale, StaticColor, PseudoColor, TrueColor, or DirectColor. (default is the default visual).

−reps <n> Specify the repetition count (Default is number that takes aprox. 5 seconds)

−subs <s0 s1 ...>
Specify the number of sub windows to use in the Window tests. Default is 4, 16, 25, 50, 75, 100 and 200.

−v1.2 Perform only x11perf Version 1.2 tests using Version 1.2 semantics.

−v1.3 Perform only x11perf Version 1.3 tests using Version 1.3 semantics.

−su Set the save_under window attribute to True on all windows created by x11perf. Default is False.

−bs <backing_store_hint>
Set the backing_store window attribute to the given value on all windows created by x11perf. <backing_store_hint> can be WhenMapped or Always. Default is NotUseful.

−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.

−oddsrect1 1x1 transparent stippled rectangle, 17x15 stipple pattern.

−oddsrect10 10x10 transparent stippled rectangle, 17x15 stipple pattern.

−oddsrect100 100x100 transparent stippled rectangle, 17x15 stipple pattern.

−oddsrect500 500x500 transparent stippled rectangle, 17x15 stipple pattern.

−oddosrect1 1x1 opaque stippled rectangle, 17x15 stipple pattern.

−oddosrect10 10x10 opaque stippled rectangle, 17x15 stipple pattern.

−oddosrect100 100x100 opaque stippled rectangle, 17x15 stipple pattern.

−oddosrect500 500x500 opaque stippled rectangle, 17x15 stipple pattern.

−oddtilerect1 1x1 tiled rectangle, 17x15 tile pattern.

−oddtilerect10 10x10 tiled rectangle, 17x15 tile pattern.

−oddtilerect100 100x100 tiled rectangle, 17x15 tile pattern.

−oddtilerect500 500x500 tiled rectangle, 17x15 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.

−trap1 Fill 1x1 trapezoid.

−trap10 Fill 10x10 trapezoid.

−trap100 Fill 100x100 trapezoid.

−trap300 Fill 300x300 trapezoid.

−strap1 Fill 1x1 transparent stippled trapezoid, 8x8 stipple pattern.

−strap10 Fill 10x10 transparent stippled trapezoid, 8x8 stipple pattern.

−strap100 Fill 100x100 transparent stippled trapezoid, 8x8 stipple pattern.

−strap300 Fill 300x300 transparent stippled trapezoid, 8x8 stipple pattern.

−ostrap1 Fill 10x10 opaque stippled trapezoid, 8x8 stipple pattern.

−ostrap10 Fill 10x10 opaque stippled trapezoid, 8x8 stipple pattern.

−ostrap100 Fill 100x100 opaque stippled trapezoid, 8x8 stipple pattern.

−ostrap300 Fill 300x300 opaque stippled trapezoid, 8x8 stipple pattern.

−tiletrap1 Fill 10x10 tiled trapezoid, 4x4 tile pattern.

−tiletrap10 Fill 10x10 tiled trapezoid, 4x4 tile pattern.

−tiletrap100 Fill 100x100 tiled trapezoid, 4x4 tile pattern.

−tiletrap300 Fill 300x300 tiled trapezoid, 4x4 tile pattern.

−oddstrap1 Fill 1x1 transparent stippled trapezoid, 17x15 stipple pattern.

−oddstrap10 Fill 10x10 transparent stippled trapezoid, 17x15 stipple pattern.

−oddstrap100 Fill 100x100 transparent stippled trapezoid, 17x15 stipple pattern.

−oddstrap300 Fill 300x300 transparent stippled trapezoid, 17x15 stipple pattern.

−oddostrap1 Fill 10x10 opaque stippled trapezoid, 17x15 stipple pattern.

−oddostrap10 Fill 10x10 opaque stippled trapezoid, 17x15 stipple pattern.

−oddostrap100 Fill 100x100 opaque stippled trapezoid, 17x15 stipple pattern.

−oddostrap300 Fill 300x300 opaque stippled trapezoid, 17x15 stipple pattern.

−oddtiletrap1 Fill 10x10 tiled trapezoid, 17x15 tile pattern.

−oddtiletrap10 Fill 10x10 tiled trapezoid, 17x15 tile pattern.

−oddtiletrap100
Fill 100x100 tiled trapezoid, 17x15 tile pattern.

−oddtiletrap300
Fill 300x300 tiled trapezoid, 17x15 tile pattern.

−bigstrap1 Fill 1x1 transparent stippled trapezoid, 161x145 stipple pattern.

−bigstrap10 Fill 10x10 transparent stippled trapezoid, 161x145 stipple pattern.

−bigstrap100 Fill 100x100 transparent stippled trapezoid, 161x145 stipple pattern.

−bigstrap300 Fill 300x300 transparent stippled trapezoid, 161x145 stipple pattern.

−bigostrap1 Fill 10x10 opaque stippled trapezoid, 161x145 stipple pattern.

−bigostrap10 Fill 10x10 opaque stippled trapezoid, 161x145 stipple pattern.

−bigostrap100 Fill 100x100 opaque stippled trapezoid, 161x145 stipple pattern.

−bigostrap300 Fill 300x300 opaque stippled trapezoid, 161x145 stipple pattern.

−bigtiletrap1 Fill 10x10 tiled trapezoid, 161x145 tile pattern.

−bigtiletrap10 Fill 10x10 tiled trapezoid, 161x145 tile pattern.

−bigtiletrap100
Fill 100x100 tiled trapezoid, 161x145 tile pattern.

−bigtiletrap300
Fill 300x300 tiled trapezoid, 161x145 tile pattern.

−eschertiletrap1
Fill 1x1 tiled trapezoid, 216x208 tile pattern.

−eschertiletrap10
Fill 10x10 tiled trapezoid, 216x208 tile pattern.

−eschertiletrap100
Fill 100x100 tiled trapezoid, 216x208 tile pattern.

−eschertiletrap300
Fill 300x300 tiled trapezoid, 216x208 tile pattern.

−complex10 Fill 10-pixel/side complex polygon.

−complex100 Fill 100-pixel/side complex polygon.

−64poly10convex
Fill 10x10 convex 64-gon.

−64poly100convex
Fill 100x100 convex 64-gon.

−64poly10complex
Fill 10x10 complex 64-gon.

−64poly100complex
Fill 100x100 complex 64-gon.

−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).

−polytext16 2-byte character in 7/14/7 line (k14, k24).

−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).

−f24itext16 2-byte character in 23-char image line (k24).

−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.

−putimagexy10
PutImage XY format 10x10 square.

−putimagexy100
PutImage XY format 100x100 square.

−putimagexy500
PutImage XY format 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.

−shmputxy10 PutImage XY format 10x10 square, MIT shared memory extension.

−shmputxy100 PutImage XY format 100x100 square, MIT shared memory extension.

−shmputxy500 PutImage XY format 500x500 square, MIT shared memory extension.

−getimage10 GetImage 10x10 square.

−getimage100 GetImage 100x100 square.

−getimage500 GetImage 500x500 square.

−getimagexy10 GetImage XY format 10x10 square.

−getimagexy100
GetImage XY format 100x100 square.

−getimagexy500
GetImage XY format 500x500 square.

−noop X protocol NoOperation.

−atom GetAtomName.

−pointer QueryPointer.

−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(7), xbench(1), x11perfcomp(1)

AUTHORS

Joel McCormack
Phil Karlton
Susan Angebranndt
Chris Kent
Keith Packard
Graeme Gill

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(7), 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*] [**−vqibn**] [*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 used to extract authorization records from one machine and merge them in on another (as is the case when using remote logins or granting 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 except when the generate command is used. Normally *xauth* is not used to create the authority file entry in the first place; *xdm* does that.

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. Use this option only to clean up stale locks.

−n This option indicates that *xauth* should not attempt to resolve any hostnames, but should simply always print the host address as stored in the authority file.

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 pair gives 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*.

generate *displayname protocolname* [**trusted|untrusted**]
[**timeout** *seconds*] [**group** *group-id*] [**data** *hexdata*]

This command is similar to add. The main difference is that instead of requiring the user to supply the key data, it connects to the server specified in *displayname* and uses the SECURITY extension in order to get the key data to store in the authorization file. If the server cannot be contacted or if it does not support the SECURITY extension, the command fails. Otherwise, an authorization entry for the indicated display using the given protocol is added to the authorization file. A protocol name consisting of just a single period is treated as an abbreviation for *MIT-*

MAGIC-COOKIE-1.

If the **trusted** option is used, clients that connect using this authorization will have full run of the display, as usual. If **untrusted** is used, clients that connect using this authorization will be considered untrusted and prevented from stealing or tampering with data belonging to trusted clients. See the SECURITY extension specification for full details on the restrictions imposed on untrusted clients. The default is **untrusted**.

The **timeout** option specifies how long in seconds this authorization will be valid. If the authorization remains unused (no clients are connected with it) for longer than this time period, the server purges the authorization, and future attempts to connect using it will fail. Note that the purging done by the server does **not** delete the authorization entry from the authorization file. The default timeout is 60 seconds.

The **group** option specifies the application group that clients connecting with this authorization should belong to. See the application group extension specification for more details. The default is to not belong to an application group.

The **data** option specifies data that the server should use to generate the authorization. Note that this is **not** the same data that gets written to the authorization file. The interpretation of this data depends on the authorization protocol. The *hexdata* is in the same format as the *hexkey* described in the add command. The default is to send no data.

[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 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 otherhost xauth merge −
```

The following command contacts the server :0 to create an authorization using the MIT-MAGIC-COOKIE-1 protocol. Clients that connect with this authorization will be untrusted.

```
% xauth generate :0 .
```

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.

HOME to get the user's home directory if XAUTHORITY isn't defined.

FILES

$HOME/.Xauthority
default authority file 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.

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.

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 your inbox in the default location for your system (some examples are /var/mail/*username*, /usr/spool/mail/*username*, /var/spool/mail/*username* (where *username* is your login name). If the MAIL environment variable is set, the file specified by it will be monitored.

−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. 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 as described above for the **−file** command line option.

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(7), xrdb(1), stat(2)

BUGS

The mailbox bitmaps are ugly.

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()**.

x^2 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()**.

y^x Raises the number on the left to the power of the number on the right. For example "2 y^x 3 =" results in "8", which is 2^3. For a further example, "(1+2+3) y^x (1+2) =" equals "6 y^x 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.

10^x 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()**.

e^x 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()
y^x	y^x	^	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/X11R6/lib/X11/app-defaults/XCalc-color*.

SEE ALSO

X(7), 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.

save Displays a File Save dialog box. Pressing the Accept button saves the currently displayed buffer to the file specified in the text field.

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  Save
                Command  next
                Command  prev
                Label  index
                Text  text
        TransientShell  fileDialogShell
                Dialog  fileDialog
                        Label  label
                        Command  accept
```

```
                    Command  cancel
                    Text value
        TransientShell  failDialogShell
            Dialog  failDialog
                    Label  label
                    Command  continue
```

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(7), 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/X11R6/lib/X11/app-defaults/XClipboard
specifies required resources

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 [**−help**] [**−analog**] [**−digital**] [**−brief**] [**−chime**] [**−hd** *color*] [**−hl** *color*] [**−update** *seconds*] [**−strftime** *format*] [**−padding** *number*] [**−norender**] [**−render**] [**−sharp**] [**−face** *pattern*]

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.

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 or **−d**
This option indicates that a 24 hour digital clock should be used.

−brief This option indicates that the digital clock should only display the hours and minutes fields. The default is to show the full time and date information.

−utime or **−d**
This option indicates that a digital clock should display seconds since the Epoch (in format ’970012340 seconds since Epoch’ instead of a standard 24-hour time.

−strftime *format*
This option allows an strftime(3) format string to be specified for the digital clock’s display.

−twelve This option indicates that a digital clock should display the time in twelve hour format.

−twentyfour
This option indicates that a digital clock should display the time in twenty-four hour format. This is the default when a digital clock is used.

−chime This option indicates that the clock should chime once on the half hour and twice on the hour.

−hands *color* (or **−hd** *color*)
This option specifies the color of the hands on an analog clock. The default is *black*. This option is effectively ignored when Xrender is in use.

−highlight *color* (or **−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*. This option is effectively ignored when Xrender is in use.

−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 30 seconds or less 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.

−render This option tells *xclock* to use the Xrender extension to draw an anti-aliased face. This is the default if *xclock* has been compiled with Xrender support. Note that the color selection options and resources used when Xrender is in effect differ from the standard options.

−norender
This option turns off the use of Xrender to draw the clock.

−sharp This option tells *xclock* to use sharper edges when drawn using the Xrender extension.

−face *pattern*
This option specifies the font to use in digital mode when the Xrender extension is used.

X DEFAULTS

This program uses the *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 tick marks. The default 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 depends on whether *reverseVideo* is specified. If *reverseVideo* is specified the default is *lwhite*, otherwise the default is *black*. Note that this resource is not used when Xrender is in effect.

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*. Note that this resource is not used when Xrender is in effect.

analog (class **Boolean)**
Specifies whether or not an analog clock should be used instead of a digital one. The default is True.

twentyfour (class **Boolean)**
Specifies whether or not a digital clock should display the time in twenty-four hour format. 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. This font is only used when Xrender is not in effect.

render (class **Boolean)**
Specifies whether or not the Xrender extension should be used for the display. The default is True if *xclock* has been compiled with Xrender support.

When Xrender is in effect, the following additional resources are understood:

face (class **FaceName)**
Specify the pattern for the font to be used for the digital clock when Xrender is used.

sharp (class **Boolean)**
Specifies if sharp edges should be used when rendering the clock. The default is False.

buffer (class **Boolean)**
Specifies that the updates of the image are drawn to a pixmap before copied into the window instead drawing them into the window directly.

The defaults of the following color resources depend on whether *reverseVideo* is specified. If *reverseVideo* is specified the default is *lwhite*, otherwise the default is *black*.

hourColor (class **Foreground)**
The color of the hour hand.

minuteColor (class **Foreground)**
The color of the minute hand.

secondColor (class **Foreground)**
The color of the second hand.

majorColor (class **Foreground)**
The color of the major scale ticks (i. e. each five minutes).

minorColor (class **Foreground)**
The color of the minor scale ticks (between major ticks).

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/X11R6/lib/X11/app-defaults/XClock
specifies required resources

SEE ALSO

X(7), xrdb(1), time(3C)

BUGS

Xclock believes the system clock.

When in digital mode, the string should be centered automatically.

AUTHORS

Tony Della Fera (MIT-Athena, DEC)
Dave Mankins (MIT-Athena, BBN)
Ed Moy (UC Berkeley)

NAME

xcmsdb − Device Color Characterization utility for X Color Management System

SYNOPSIS

xcmsdb [**−query**] [**−remove**] [**−format 32|16|8**] [*filename*]

DESCRIPTION

xcmsdb is used to load, query, or remove Device Color Characterization data stored in properties on the root window of the screen as specified in section 7, Device Color Characterization, of the ICCCM. Device Color Characterization data (also called the Device Profile) is an integral part of Xlib's X Color Management System (Xcms), necessary for proper conversion of color specification between device-independent and device-dependent forms. Xcms uses 3x3 matrices stored in the XDCCC_LINEAR_RGB_MATRICES property to convert color specifications between CIEXYZ and RGB Intensity (XcmsRGBi, also referred to as linear RGB). Xcms then uses display gamma information stored in the XDCCC_LINEAR_RGB_CORRECTION property to convert color specifications between RGBi and RGB device (XcmsRGB, also referred to as device RGB).

Note that Xcms allows clients to register *function sets* in addition to its built-in function set for CRT color monitors. Additional function sets may store their device profile information in other properties in function set specific format. This utility is unaware of these non-standard properties.

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.

−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.

AUTHOR

Chuck Adams, Tektronix Inc. Al Tabayoyon, SynChromatics Inc. (added multi-visual support)

NAME

xconsole − monitor system console messages with X

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.

−saveLines *count*
: When set, *xconsole* only preserves *count* lines of message history instead of growing the text buffer without bound (a *count* of zero − the default − is treated as placing no limit on the history).

X DEFAULTS

This program uses the *Athena Text* widget, look in the *Athena Widget Set* documentation for controlling it.

Xconsole otherwise accepts resources of the same names as the command-line options (without the leading dash). "file" is a string type, "saveLines" an integer, and the remaining options are booleans.

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/X11R6/lib/X11/app-defaults/XConsole
: specifies required resources

SEE ALSO

X(7), xrdb(1), Athena Text widget

AUTHOR

Keith Packard (MIT X Consortium)

NAME

xcursorgen − create an X cursor file from a collection of PNG images

SYNOPSIS

xcursorgen [*config-file*] [*output-file*]

DESCRIPTION

Xcursorgen reads the config-file to find the list of cursor images along with their hotspot and nominal size information. Xcursorgen converts all of the images to Xcursor format and writes them to the output-file.

Each line in the config file is of the form:
<size> <xhot> <yhot> <filename> <ms-delay>

Multiple images with the same <size> are used to create animated cursors, the <ms-delay> value on each line indicates how long each image should be displayed before switching to the next. <ms-delay> can be elided for static cursors.

SEE ALSO

Xcursor(3x)

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(7), 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.

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 suppresses 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 new-line 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(7), *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, host chooser

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 Open Group 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.

Xdm can be configured to ignore BroadcastQuery messages from selected hosts. This is useful when you don't want the host to appear in menus produced by *chooser* or X terminals 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.

OVERVIEW

xdm is highly configurable, and most of its behavior can be controlled by resource files and shell scripts. The names of these files themselves are resources read from the file *xdm-config* or the file named by the **−config** option.

xdm offers display management two different ways. It can manage X servers running on the local machine and specified in *Xservers*, and it can manage remote X servers (typically X terminals) using XDMCP (the XDM Control Protocol) as specified in the *Xaccess* file.

The resources of the X clients run by *xdm* outside the user's session, including *xdm*'s own login window, can be affected by setting resources in the *Xresources* file.

For X terminals that do not offer a menu of hosts to get display management from, *xdm* can collect willing hosts and run the *chooser* program to offer the user a menu. For X displays attached to a host, this step is typically not used, as the local host does the display management.

After resetting the X server, *xdm* runs the *Xsetup* script to assist in setting up the screen the user sees along with the *xlogin* widget.

The *xlogin* widget, which *xdm* presents, offers the familiar login and password prompts.

After the user logs in, *xdm* runs the *Xstartup* script as root.

Then *xdm* runs the *Xsession* script as the user. This system session file may do some additional startup and typically runs the *.xsession* script in the user's home directory. When the *Xsession* script exits, the session is over.

At the end of the session, the *Xreset* script is run to clean up, the X server is reset, and the cycle starts over.

The file */usr/X11R6/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.

OPTIONS

All of these options, except **−config** itself, 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/X11R6/lib/X11/xdm/xdm-config* is the default. See the section **Configuration File**.

−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 **Local 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. If set to 0 xdm will not listen for XDMCP or Chooser requests.

−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 local displays, the resource name and class are as read from the *Xservers* file.

For remote displays, the resource name is what the network address of the display resolves to. See the **removeDomain** resource. The name must match exactly; *xdm* is not aware of all the network aliases that might reach a given display. If the name resolve fails, the address is used. The resource class is as sent by the display in the XDMCP Manage request.

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. For example, **DisplayManager.expo_x_org_0.startup** is the name of the resource which defines the startup shell file for the ''expo.x.org:0'' display.

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 **Local 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 under which *xdm* stores authorization files while initializing the session. The default value is */usr/X11R6/lib/X11/xdm.* Can be overridden for specific displays by DisplayManager.*DISPLAY*.authFile.

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 list of additional environment variables, separated by white space, 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.prngdSocket

DisplayManager.prngPort
A UNIX domain socket name or a TCP socket port number on local host on which a Pseudo-Random Number Generator Daemon, like EGD (http://egd.sourceforge.net) is listening, in order to generate the autorization keys. Either a non null port or a valid socket name must be specified. The default is to use the Unix-domain socket */tmp/entropy*.

On systems that don't have such a daemon, a fall-back entropy gathering system, based on various log file contents hashed by the MD5 algorithm is used instead.

DisplayManager.greeterLib
On systems that support a dynamically-loadable greeter library, the name of the library. The default is
/usr/X11R6/lib/X11/xdm/libXdmGreet.so.

DisplayManager.choiceTimeout
Number of seconds to wait for display to respond after user has selected a host from the chooser. If the display sends an XDMCP IndirectQuery within this time, the request is forwarded to the chosen host. Otherwise, it is assumed to be from a new session and the chooser is offered again. Default is 15.

DisplayManager.sourceAddress
Use the numeric IP address of the incoming connection on multihomed hosts instead of the host name. This is to avoid trying to connect on the wrong interface which might be down at this time.

DisplayManager.willing
This specifies a program which is run (as) root when an an XDMCP BroadcastQuery is received and this host is configured to offer XDMCP display management. The output of this program may be displayed on a chooser window. If no program is specified, the string *Willing to manage* is sent.

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/X11R6/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.
CHOOSERPATH 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/X11R6/bin/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 appearance 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/X11R6/bin/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. 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/X11R6/bin:/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/X11R6/bin:/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/X11R6/bin/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 list of authorization mechanisms to use, separated by white space. 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,’’ or, if XDM-AUTHORIZATION-1 is available, ‘‘XDM-AUTHORIZATION-1 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. If not specified, a name is generated from DisplayManager.authDir and the name of the display.

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.

CONFIGURATION FILE

First, the *xdm* configuration file should be set up. Make a directory (usually */usr/X11R6/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/X11R6/lib/X11/xdm/Xservers
DisplayManager.errorLogFile:      /usr/X11R6/lib/X11/xdm/xdm-errors
DisplayManager*resources:         /usr/X11R6/lib/X11/xdm/Xresources
DisplayManager*startup:           /usr/X11R6/lib/X11/xdm/Xstartup
DisplayManager*session:           /usr/X11R6/lib/X11/xdm/Xsession
DisplayManager.pidFile:           /usr/X11R6/lib/X11/xdm/xdm-pid
DisplayManager._0.authorize:      true
DisplayManager*authorize:         false
```

Note that this file mostly 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.

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.

To only respond to Direct queries for a host or pattern, it can be followed by the optional ‘‘NOBROADCAST’’ keyword. This can be used to prevent an xdm server from appearing on menus based on Broadcast queries.

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

*.deshaw.com              NOBROADCAST              # allow only direct access
*.gw.com                                           # allow direct and broadcast

#
# 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
```

If compiled with IPv6 support, multicast address groups may also be included in the list of addresses indirect queries are set to. Multicast addresses may be followed by an optional / character and hop count. If no hop count is specified, the multicast hop count defaults to 1, keeping the packet on the local network.

For IPv4 multicasting, the hop count is used as the TTL.

Examples:

```
rincewind.sample.net    ff02::1                     #IPv6 Multicast to ff02::1
                                                    #with a hop count of 1
ponder.sample.net       CHOOSER 239.192.1.1/16  #Offer a menu of hosts
                                                    #who respond to IPv4 Multicast
                                                    # to 239.192.1.1 with a TTL of 16
```

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. For more flexibility at this step, the chooser could be a shell script. *Chooser* is the session manager here; it is run instead of a child *xdm* to manage the display.

Resources for this program can be put into the file named by **DisplayManager.***DISPLAY***.resources**.

When the user selects a host, *chooser* prints the host chosen, which is read by the parent *xdm*, and exits. *xdm* closes its connection to the X server, and the server resets and sends another **Indirect** XDMCP request. *xdm* remembers the user’s choice (for **DisplayManager.choiceTimeout** seconds) and forwards the request to the chosen host, which starts a session on that display.

LISTEN

The following configuration directive is also defined for the Xaccess configuration file:

LISTEN *interface [list of multicast group addresses]*
: *interface* may be a hostname or IP addresss representing a network interface on this machine, or the wildcard * to represent all available network interfaces.

If one or more LISTEN lines are specified, xdm only listens for XDMCP connections on the specified interfaces. If multicast group addresses are listed on a listen line, xdm joins the multicast groups on the given interface.

If no LISTEN lines are given, the original behavior of listening on all interfaces is preserved for backwards compatibility. Additionally, if no LISTEN is specified, xdm joins the default XDMCP IPv6 multicast group, when compiled with IPv6 support.

To disable listening for XDMCP connections altogther, a line of LISTEN with no addresses may be specified, or the previously supported method of setting DisplayManager.requestPort to 0 may be used.

Examples:

```
LISTEN * ff02::1            # Listen on all interfaces and to the
                            # ff02::1 IPv6 multicast group.
LISTEN 10.11.12.13          # Listen only on this interface, as long
                            # as no other listen directives appear in
                            # file.
```

IPv6 MULTICAST ADDRESS SPECIFICATION

The Internet Assigned Numbers Authority has has assigned ff0*X*:0:0:0:0:0:0:12b as the permanently assigned range of multicast addresses for XDMCP. The *X* in the prefix may be replaced by any valid scope identifier, such as 1 for Node-Local, 2 for Link-Local, 5 for Site-Local, and so on. (See IETF RFC 2373 or its replacement for further details and scope definitions.) xdm defaults to listening on the Link-Local scope address ff02:0:0:0:0:0:0:12b to most closely match the old IPv4 subnet broadcast behavior.

LOCAL 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. This method is used typically for local servers only. If the resource or the file named by the resource is empty, *xdm* will offer XDMCP service only.

Each specification 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/X11R6/bin/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 Sun-CG3 local /usr/X11R6/bin/X :0" instead of "localhost:0 Sun-CG3 local /usr/X11R6/bin/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 (such as 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.

When *xdm* starts a session, it sets up authorization data for the server. For local servers, *xdm* passes "**−auth** *filename*" on the server's command line to point it at its authorization data. For XDMCP servers, *xdm* passes the authorization data to the server via the **Accept** XDMCP request.

RESOURCES FILE

The *Xresources* file 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\
        Ctrl<Key>R: abort-display()\n\
        <Key>F1: set-session-argument(failsafe) finish-field()\n\
        <Key>Return: set-session-argument() finish-field()
xlogin*borderWidth: 3
xlogin*greeting: CLIENTHOST
#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).

This file may also contain resources for the setup program and *chooser*.

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**.

Here is a sample *Xsetup* script:

```
#!/bin/sh
# Xsetup_0 − setup script for one workstation
xcmsdb < /usr/X11R6/lib/monitors/alex.0
xconsole −geometry 480x130−0−0 −notify −verbose −exitOnFail &
```

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.allowRootLogin
If set to ‘‘false’’, don’t allow root (and any other user with uid = 0) to log in directly. The default is ‘‘true’’.

xlogin.Login.allowNullPasswd
If set to ‘‘true’’, allow an otherwise failing password match to succeed if the account does not require a password at all. The default is ‘‘false’’, so only users that have passwords assigned can log in.

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 action is not accessible in the default configuration. There are various reasons to stop *xdm* on a system console, such as when shutting the system down, when using *xdmshell*, to start another type of server, or to generally access the console. Sending *xdm* a SIGHUP will restart the display. See the section **Controlling XDM**.

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 section **Session Program**.

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.

On some systems (OpenBSD) the user's shell must be listed in */etc/shells* to allow login through xdm. The normal password and account expiration dates are enforced too.

STARTUP PROGRAM

The *Xstartup* program is run as root when the user logs in. It is typically a shell script. Since it is run as root, *Xstartup* should be very careful about security. This is the place to put commands which add entries to */etc/utmp* (the *sessreg* program may be useful here), mount users' home directories from file servers, 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
LOGNAME	the user name

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.

The sample *Xstartup* file shown here prevents login while the file */etc/nologin* exists. 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
        xmessage −file /etc/nologin −timeout 30 −center
        exit 1
fi
sessreg −a −l $DISPLAY −x /usr/X11R6/lib/xdm/Xservers $LOGNAME
/usr/X11R6/lib/xdm/GiveConsole
exit 0
```

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
LOGNAME	the user name
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
KRB5CCNAME	may be set to a Kerberos credentials cache name

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 example following demonstrates this feature.

This example recognizes the special ''failsafe'' mode, specified in the translations in the *Xresources* file, to provide an escape from the ordinary session. It also requires that the .xsession file be executable so we don't have to guess what shell it wants to use.

```
#!/bin/sh
#
# Xsession
#
# This is the program that is run as the client
# for the display manager.

case $# in
1)
        case $1 in
        failsafe)
                exec xterm −geometry 80x24−0−0
                ;;
        esac
esac

startup=$HOME/.xsession
resources=$HOME/.Xresources

if [ −f "$startup" ]; then
        exec "$startup"
else
        if [ −f "$resources" ]; then
                xrdb −load "$resources"
        fi
        twm &
        xman −geometry +10−10 &
        exec xterm −geometry 80x24+10+10 −ls
fi
```

The user's *.xsession* file might look something like this example. Don't forget that the file must have execute permission.

```
#! /bin/csh
# no −f in the previous line so .cshrc gets run to set $PATH
twm &
xrdb −merge "$HOME/.Xresources"
emacs −geometry +0+50 &
xbiff −geometry −430+5 &
xterm −geometry −0+50 -ls
```

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*.

A sample *Xreset* script:

```
#!/bin/sh
#
# Xreset
#
# This program is run as root after the session ends
#
sessreg −d −l $DISPLAY −x /usr/X11R6/lib/xdm/Xservers $LOGNAME
/usr/X11R6/lib/xdm/TakeConsole
```

```
exit 0
```

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.*

ADDITIONAL LOCAL DISPLAYS

To add an additional local display, add a line for it to the *Xservers* file. (See the section **Local Server Specification**.)

Examine the display-specific resources in *xdm-config* (e.g., **DisplayManager._0.authorize**) and consider which of them should be copied for the new display. The default *xdm-config* has all the appropriate lines for displays **:0** and **:1**.

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/X11R6/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/X11R6/lib/X11/xdm/xdm-config
the default configuration file

$HOME/.Xauthority user authorization file where *xdm* stores keys for clients to read

CHOOSERPATH the default chooser

/usr/X11R6/bin/xrdb the default resource database loader

/usr/X11R6/bin/X the default server

/usr/X11R6/bin/xterm the default session program and failsafe client

/usr/X11R6/lib/X11/xdm/A<display>−<suffix>
the default place for authorization files

/tmp/K5C<display> Kerberos credentials cache

SEE ALSO

X(7), *xinit*(1), *xauth*(1), *Xsecurity*(7), *sessreg*(1), *Xserver*(1),
X Display Manager Control Protocol

AUTHOR

Keith Packard, MIT X Consortium

NAME

xdpyinfo − display information utility for X

SYNOPSIS

xdpyinfo [−display *displayname*] [−queryExtensions] [−ext *extension-name*]

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.

By default, numeric information (opcode, base event, base error) about protocol extensions is not displayed. This information can be obtained with the **−queryExtensions** option. Use of this option on servers that dynamically load extensions will likely cause all possible extensions to be loaded, which can be slow and can consume significant server resources.

Detailed information about a particular extension is displayed with the **−ext** *extensionName* option. If *extensionName* is **all**, information about all extensions supported by both *xdpyinfo* and the server is displayed.

ENVIRONMENT

DISPLAY

To get the default host, display number, and screen.

SEE ALSO

X(7), xwininfo(1), xprop(1), xrdb(1)

AUTHOR

Jim Fulton, MIT X Consortium
Support for the XFree86-VidModeExtension, XFree86-DGA, XFree86-Misc, and XKB extensions added by Joe Moss

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 also 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*(7)). The order of the command line options is not important.

filename

Specifies the file(s) that are to be loaded during start-up. This is the file which will be edited. If a file is not specified, *xedit* lets you load files or create new files 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 has the input focus, or the Text widget that the pointer cursor is currently over.

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	Keyboard 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 [*number*]	Multiply by 4 or *number*	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	Meta-z	Scroll One Line Down
Ctrl-_	Undo		
Escape	Line Edit Mode		

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)

LINE EDIT MODE

Line edit mode enables several shortcut commands for searching and replacing text in a xedit buffer. *Line edit mode* commands have the format:

[*line-number*[,*line-number*]]*command*[*parameters*]

Line number may be specified as:

. The current text line.

$ The last line of the file.

number The literal line *number*.

- or ^ The previous line. Equivalent to *-1*.

-number or ^number
The current line minus *number*.

+ The next line. Equivalent to *+1*.

+number
The current line plus *number*.

, or % From the first to the last line. Equivalent to *1,$*.

; From the current to the last line. Equivalent to *.,$*.

Command may be specified as:

s Substitute text in the specified lines.

/re/ Search forward for the regular expression pattern *re*.

?re? Search backward for the regular expression pattern *re*.

Parameters may be specified as:

/re/ Works as a parameter to *i* or as a command.

/re/text/ Search forward for *re* and substitute by *text*.

Options may follow or be parameters, known values are:

i Case insensitive search.

g *Global* match when replacing text. Unless specified, only the nth, that defaults to 1, match will be replaced.

c *Confirm* before replacing text.

number Replace only the occurrence referenced by *number*.

Commands accept some variations, examples:

/pattern/i
i/pattern/
i/pattern

Search forward for *pattern*.

,sc/pattern/text
,sc/pattern/text/
,s/pattern/text/c

Search the entire buffer and ask confirmation to replace *pattern* with *text*.

,s/pattern/text/number

Replace the match *number* in the text line. If not specified, defaults to the first occurrence.

When searching for text, type <Return> to go to the next match. When interactively replacing text, type *y* or *Y* to accept the change, and *n* or *N* to ignore it and go to the next match.

COMMANDS

Quit Quits the current editing session. If any changes have not been saved, *xedit* displays a warning message, allowing the user to save them.

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.

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 ''on,'' stating that 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 ''~'' as the suffix.

positionFormat (Class **Format)**
Specifies a format string used to display the cursor position. This string uses printf(3) like notation, where **%l** prints the line number, **%c** prints the column number, **%p** prints the insert position offset, and **%s** prints the current file size. It is also allowed to specify field sizes, with the notation **%−?[0−9]+** . The default format string is ''L%l'', which shows the character ''L'' followed by the line number.

hints (Class **Hints)**
Specifies a list of strings, separated by new lines, that will be displayed in the bc_label window.

hintsInterval (Class **Interval)**
Specifies the interval in seconds, which the hint string in the bc_label window will be changed.

changedBitmap (Class **Bitmap)**
Specifies the name of the Bitmap that will be displayed in the fileMenu, when the file being edited is changed.

autoReplace (Class **Replace)**
This resource is useful to automatically correct common misspelling errors, but can also be used to create simple macros. The format is *{non-blanks}{blanks}[{string}]*. Fields are separeted by newlines. Example of use:

```
nto        not\n\
/macro some long string with \\\n newlines \\\n
```

Will automatically replace the word *nto* by *not*, and */macro* by *some long string with newlines* when you type that words.

ispell.dictionaries (Class **ispell.Dictionary**)
Specifies a list of dictionary names, separeted by spaces, available to the ispell program. The default value is *"american americamed+ english"*.

ispell.dictionary (Class **ispell.Dictionary**)
Specifies the default dictionary to use.

ispell*<DICTIONARY>.wordChars (Class **ispell*Chars)**
Specifies a set of characters that can be part of a legal word. The *<DICTIONARY>* field is one of the dictionaries specified in the *dictionaries* resource.

ispell.ispellCommand (Class **ispell.CommandLine)**
The path to the ispell program, and possibly, additional arguments. You don't need to specify the ‘‘-w’’ option, neither the ‘‘-a’’ option. Refer to the *ispell*(1) manpage for more information on ispell options.

ispell.formatting (Class **ispell.TextFormat)**
Specifies which text formatting to use while spell checking the file. The available formats are *text* and *html*.

ispell*text.skipLines (Class **ispell*text.Skip)**
Lines starting with one of the characters in this string will not be spell checked. This resource is only used in *text* mode.

ispell.terseMode (Class **ispell.Terse)**
When enabled, runs ispell in terse mode, not asking user interaction for words generated through compound formation (when using the ispell ‘‘-C’’ option), or words generated through affix removal. The default value is False.

ispell.lookCommand (Class **ispell.CommandLine)**
The path to the program to search for alternate words, and possibly, additional arguments. The default program used is */usr/bin/egrep*.

ispell.wordsFile (Class **ispell.Words)**
The path to the file[s] to search for alternate words. The default file is */usr/share/dict/words*.

ispell.guessLabel (Class **ispell.Status)**
String displayed in the ispell status bar when ispell returns a guess list of one or more words. The default value is *Guess*.

ispell.missLabel (Class **ispell.Status)**
String displayed in the ispell status bar when ispell returns a list of one or more words to match a misspelled one. The default value is *Miss*.

ispell.rootLabel (Class **ispell.Status)**
String displayed in the ispell status bar when the word is not in the dictionary, but it can be formed through a root one. The default value is *Root:*, and is followed by a space and the root

word.

ispell.noneLabel (Class **ispell.Status)**
String displayed in the ispell status bar when there is no near misses. The default value is *None*.

ispell.compoundLabel (Class **ispell.Status)**
String displayed in the ispell status bar when the word being checked is formed by concatenation of two words. The default value is *Compound*.

ispell.okLabel (Class **ispell.Status)**
String displayed in the ispell status bar when the checked word is in the dictionary. This string is only displayed when using the *check* button in the xedit ispell interface. The default value is *Ok*.

ispell.eofLabel (Class **ispell.Status)**
The string displayed in the ispell status bar when the end of the file is reached. The default value is *End Of File*.

ispell.repeatLabel (Class **ispell.Status)**
The string displayed in the ispell status bar when two identical words are found together in the file. The default value is *Repeat*.

ispell.lookLabel (Class **ispell.Status)**
The string displayed in the ispell status bar after displaying the results of the *Look* command. If no results are found, the value of the *ispell.noneLabel* resource is shown.

ispell.workingLabel (Class **ispell.Status)**
The string displayed in the ispell status bar while xedit is communicating with ispell. The default value is ....

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/X11R6/lib/X11/app-defaults/Xedit
specifies required resources

SEE ALSO

X(7), *xrdb*(1), *Athena Widget Set*

RESTRICTIONS

Xedit is not a replacement to Emacs.

COPYRIGHT

AUTHORS

Chris D. Peterson, MIT X Consortium
Paulo Cesar Pereira de Andrade, The XFree86 Project

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 it 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; it is essentially a debugging and development tool, and should not be needed in normal usage.

OPTIONS

−display *display*
This option specifies the X server to contact.

−geometry *geom*
This option specifies the size and/or location of the window, if a window is to be created.

−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. Backing store refers to the the pixels saved off-screen when the X server maintains the contents of a window; NotUseful means that the xev process will redraw its contents itself, as necessary.

−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. Save unders are similar to backing store, but they refer rather to the saving of pixels off-screen when the current window obscures other windows. Save unders are only advisory, and are normally set for popup dialogs and other transient windows.

−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(7), xwininfo(1), xdpyinfo(1), Xlib Programmers Manual, X Protocol Specification
See *X(7)* for a full statement of rights and permissions.

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(7).*

−display *display*
specify the display to use; see *X(7).*

−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(7), X Toolkit documentation
See *X(7)* for a full statement of rights and permissions.

AUTHOR

Keith Packard, MIT X Consortium
Copied from the NeWS version written (apparently) by Jeremy Huxtable as seen at SIGGRAPH ’88

NAME

xfd − display all the characters in an X font

SYNOPSIS

xfd [−options ...] **−fn** *fontname*

xfd [−options ...] **−fa** *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 clicking 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 core X server side font to be displayed. This can also be set with the FontGrid **font** resource. A font must be specified.

−fa *font* This option specifies a Xft font to be displayed. This can also be set with the FontGrid **face** resource. A font pattern must be specified.

−box This option indicates that a box should be displayed outlining the area that would be filled with background color by an ImageText request. This can also be set with the FontGrid **boxChars** resource. The default is False.

−center This option indicates that each glyph should be centered in its grid. This can also be set with the FontGrid **centerChars** resource. The default is False.

−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. This can also be set with the FontGrid **startChar** resource. The default is 0.

−bc *color*
This option specifies the color to be used if ImageText boxes are drawn. This can also be set with the FontGrid **boxColor** resource.

−rows *numrows*
This option specifies the number of rows in the grid. This can also be set with the FontGrid **cellRows** resource.

−columns *numcols*

This option specifies the number of columns in the grid. This can also be set with the FontGrid **cellColumns** resource.

WIDGETS

In order to specify resources, it is useful to know the widgets which compose *xfd*. 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 Xfd.

```
Xfd  xfd
        Paned  pane
                Label  fontname
                Box  box
                        Command  quit
                        Command  prev
                        Command  next
                Label  select
                Label  metrics
                Label  range
                Label  start
                Form  form
                        FontGrid  grid
```

FONTGRID RESOURCES

The FontGrid widget is an application-specific widget, and a subclass of the Simple widget in the Athena widget set. The effects and instance names of this widget's resources are given in the **OPTIONS** section. Capitalize the first letter of the resource instance name to get the corresponding class name.

APPLICATION SPECIFIC RESOURCES

The instance names of the application specific resources are given below. Capitalize the first letter of the resource instance name to get the corresponding class name. These resources are unlikely to be interesting unless you are localizing xfd for a different language.

selectFormat

Specifies a printf-style format string used to display information about the selected character. The default is "character 0x%02x%02x (%u,%u) (%#o,%#o)". The arguments that will come after the format string are char.byte1, char.byte2, char.byte1, char.byte2, char.byte1, char.byte2. char.byte1 is byte 1 of the selected character. char.byte2 is byte 2 of the selected character.

metricsFormat

Specifies a printf-style format string used to display character metrics. The default is "width %d; left %d, right %d; ascent %d, descent %d (font %d, %d)". The arguments that will come after the format string are the character metrics width, lbearing, rbearing, character ascent, character descent, font ascent, and font descent.

rangeFormat

Specifies a printf-style format string used to display the range of characters currently being displayed. The default is "range: 0x%02x%02x (%u,%u) thru 0x%02x%02x (%u,%u)". The arguments that will come after the format string are the following fields from the XFontStruct that is returned from opening the font: min_byte1, min_char_or_byte2, min_byte1, min_char_or_byte2, max_byte1, max_char_or_byte2, max_byte1, max_char_or_byte2.

startFormat

Specifies a printf-style format string used to display information about the character at the upper left corner of the font grid. The default is "upper left: 0x%04x (%d,%d)". The arguments that will come after the format string are the new character, the high byte of the new character, and the low byte of the new character.

nocharFormat
Specifies a printf-style format string to display when the selected character does not exist. The default is "no such character 0x%02x%02x (%u,%u) (%#o,%#o)". The arguments that will come after the format string are the same as for the **selectFormat** resource.

SEE ALSO

X(7), xlsfonts(1), xrdb(1), xfontsel(1), fontconfig(3), *X Logical Font Description Conventions*

BUGS

The program should skip over pages full of non-existent characters.

AUTHOR

Jim Fulton, MIT X Consortium; previous program of the same name by Mark Lillibridge, MIT Project Athena.

NAME

xfindproxy - locate proxy services

SYNOPSIS

xfindproxy −manager *managerAddr* **−name** *serviceName* **−server** *serverAddr* [**−auth**] [**−host** *hostAddr*] [**−options** *opts*]

DESCRIPTION

xfindproxy is a program used to locate available proxy services. It utilizes the Proxy Management Protocol to communicate with a proxy manager. The proxy manager keeps track of all available proxy services, starts new proxies when necessary, and makes sure that proxies are shared whenever possible.

The **−manager** argument is required, and it specifies the network address of the proxy manager. The format of the address is a standard ICE network id (for example, "tcp/blah.x.org:6500").

The **−name** argument is required, and it specifies the name of the desired proxy service (for example, "LBX"). The name is case insensitive.

The **−server** argument is also required, and it specifies the address of the target server. The format of the address is specific to the proxy service specified with the -name argument. For example, for a proxy service of "LBX", the address would be an X display address (e.g, "blah.x.org:0").

The **−auth** argument is optional. If specified, xfindproxy will read 2 lines from standard input. The first line is an authorization/authentication name. The second line is the authorization/authentication data in hex format (the same format used by xauth). xfindproxy will pass this auth data to the proxy, and in most cases, will be used by the proxy to authorize/authenticate itself to the target server.

The **−host** argument is optional. If xfindproxy starts a new proxy service, it will pass the host specified. The proxy may choose to restrict all connections to this host. In the event that xfindproxy locates an already existing proxy, the host will be passed, but the semantics of how the proxy uses this host are undefined.

The **−options** argument is optional. If xfindproxy starts a new proxy service, it will pass any options specified. The semantics of the options are specific to each proxy server and are not defined here. In the event that xfindproxy locates an already existing proxy, the options will be passed, but the semantics of how the proxy uses these options are undefined.

If xfindproxy is successful in obtaining a proxy address, it will print it to stdout. The format of the proxy address is specific to the proxy service being used. For example, for a proxy service of "LBX", the proxy address would be the X display address of the proxy (e.g, "blah.x.org:63").

If xfindproxy is unsuccessful in obtaining a proxy address, it will print an error to stderr.

SEE ALSO

proxymngr (1), Proxy Management Protocol spec V1.0

AUTHOR

Ralph Mor, X Consortium

NAME

xfontsel – point and click selection of X11 font names

SYNTAX

xfontsel [-*toolkitoption* ...] **[-pattern** *fontname***] [-print] [-sample** *text***] [-sample16** *text16***] [-sampleUCS** *textUCS***] [-scaled]**

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 **-sampleUCS** option specifies the sample text encoded in the UTF-8 form to be used to display the selected font if the font has a CHARSET_REGISTRY of ISO10646, overriding the default.

The **-scaled** option enables the ability to select scaled fonts at arbitrary pixel or point sizes.

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 */usr/X11R6/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
Mark Leisher <mleisher@crl.nmsu.edu> added the support for the UTF-8 sample text.

NAME

xfs − X font server

SYNOPSIS

xfs [−config *configuration_file*] [−daemon] [−droppriv] [−ls *listen_socket*] [−nodaemon] [−port *tcp_port*] [−user *username*]

DESCRIPTION

Xfs 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. If this parameter is not specified, the default file, */usr/X11R6/lib/X11/fs/config* will be used.

−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. The default port number is 7100.

−daemon
Instructs xfs to fork and go into the background automatically at startup If this option is not specified, xfs will run as a regular process (unless xfs was built to daemonize by default).

−droppriv
If specified, xfs will attempt to run as user and group *xfs* (unless the **−user** option is used). This has been implemented for security reasons, as xfs may have undiscovered buffer overflows or other paths for possible exploit, both local and remote. With this option, you may also wish to specify "no-listen = tcp" in the config file, which ensures that xfs will not to use a TCP port at all.

−nodaemon
When xfs is built to daemonize (run in the background) by default, this prevents that and starts xfs up as a regular process.

−user username
This is equivalent to **−droppriv** except that xfs will run as user *username*.

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:

catalogue (list of string)
Ordered list of font path element names. Use of the keyword "catalogue" is very misleading at present, the current implementation only supports a single catalogue ("all"), containing all of the specified fonts.

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. The default is 120.

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. A resolution is a comma-separated pair of x and y resolutions in pixels per inch. Multiple resolutions are separated by commas.

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

no-listen (trans-type)
Disable a transport type. For example, TCP/IP connections can be disabled with no-listen tcp

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.

deferglyphs (string)
Set the mode for delayed fetching and caching of glyphs. Value is "none", meaning deferred glyphs is disabled, "all", meaning it is enabled for all fonts, and "16", meaning it is enabled only for 16-bits fonts.

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:7101,hansen:7102

# 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/X11R6/lib/X11/fonts/speedo,
        /usr/X11R6/lib/X11/fonts/misc,
```

```
        /usr/X11R6/lib/X11/fonts/100dpi/

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

# 100 x 100 and 75 x 75
default-resolutions = 100,100,75,75
use-syslog = off
```

FONT SERVER NAMES

One of the following forms can be used to name a font server that accepts TCP connections:

tcp/*hostname*:*port*
tcp/*hostname*:*port*/*cataloguelist*

The *hostname* specifies the name (or decimal numeric address) of the machine on which the font server is running. The *port* is the decimal TCP port on which the font server is listening for connections. The *cataloguelist* specifies a list of catalogue names, with '+' as a separator.

Examples: *tcp/fs.x.org:7100*, *tcp/18.30.0.212:7101/all*.

One of the following forms can be used to name a font server that accepts DECnet connections:

decnet/*nodename*::font$*objname*
decnet/*nodename*::font$*objname*/*cataloguelist*

The *nodename* specifies the name (or decimal numeric address) of the machine on which the font server is running. The *objname* is a normal, case-insensitive DECnet object name. The *cataloguelist* specifies a list of catalogue names, with '+' as a separator.

Examples: *DECnet/SRVNOD::FONT$DEFAULT*, *decnet/44.70::font$special/symbols*.

SEE ALSO

X(7), *The X Font Service Protocol*,
Font server implementation overview

BUGS

Multiple catalogues should be supported.

AUTHORS

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

NAME

xfsinfo − X font server information utility

SYNOPSIS

xfsinfo [−server *servername*]

DESCRIPTION

Xfsinfo 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.

OPTIONS

−server *host***:***port*

This option specifies the X font server to contact.

HISTORY

Xfsinfo was originally called *fsinfo.* It was renamed to avoid a clash with the *fsinfo* utility from the Berkeley automounter *amd.*

EXAMPLE

The following shows a sample produced by *xfsinfo.*

```
name of server:    hansen:7100
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:7101
    #1    hansen:7102
number of extensions:      0
```

ENVIRONMENT

FONTSERVER

To get the default fontserver.

SEE ALSO

xfs(1), fslsfonts(1)

AUTHOR

Dave Lemke, Network Computing Devices, Inc

NAME

xfwp - X firewall proxy

SYNOPSIS

xfwp [option ...]

COMMAND LINE OPTIONS

The command line options that can be specified are:

–cdt *num_secs*
Used to override the default time-to-close (604800 seconds) for xfwp client data connections on which there is no activity (connections over which X protocol is already being relayed by xfwp)

–clt *num_secs*
Used to override the default time-to-close (86400 seconds) for xfwp client listen ports (ports on xfwp to which X clients first connect when trying to reach an X server)

–pdt *num_secs*
Used to override the default time-to-close (3600 seconds) for Proxy Manager connections on which there is no activity

–config *file_name*
Used to specify the configuration the name of the configuration file

–pmport *port_number*
Used to override the default port address (4444) for proxy manager connections

–verify Used to display the configuration file rule that was actually matched for each service request

–logfile *file_name*
Used to specify the name of a file where audit information should be logged. The format of a logged entry is: time of day; event code; source IP address; destination IP address; and configuration rule number. The event codes are: "0" for a successful connection; "1" if a connection is denied because of a configuration rule; and "2" if a connection is denied because of an authorization failure. If the event code is "1", and a configuration file is used, the configuration rule number is the line number of the configuration file where the match was made (see the section CONFIGURATION FILE for more information). If the event code is not "1", or if no configuration file is used, the configuration rule number is "-1".

–loglevel *{0,1}*
Used to specify the amount of audit detail that should be logged. If "0", all connections are logged. If "1", only unsuccessful connections are logged.

–max_pm_conns *num_connections*
Used to specify the maximum number of Proxy Manager connections. The default is 10.

–max_pm_conns *num_connections*
Used to specify the maximum number of X server connections. The default is 100.

DESCRIPTION

The X firewall proxy (xfwp) is an application layer gateway proxy that may be run on a network firewall host to forward X traffic across the firewall. Used in conjunction with the X server Security extension and authorization checking, xfwp constitutes a safe, simple, and reliable mechanism both to hide the addresses of X servers located on the Intranet and to enforce a server connection policy. Xfwp cannot protect against mischief originating on the Intranet; however, when properly configured it can guarantee that only trusted clients originating on authorized external Internet hosts will be allowed inbound access to local X servers.

To use xfwp there must be an X proxy manager running in the local environment which has been configured at start-up to know the location of the xfwp. [NOTE: There may be more than one xfwp running in a local environment; see notes below on load balancing for further discussion.] Using the xfindproxy utility (which relays its requests through the proxy manager) a user asks xfwp to allocate a client listen port for a particular X server, which is internally associated with all future connection requests for that server. This client listen port address is returned by the proxy manager through xfindproxy. The xfwp hostname and port number is then passed out-of-band (i.e., via a Web browser) to some remote X client, which will subsequently connect to xfwp instead of to the target X server.

When an X client connection request appears on one of xfwp's listen ports, xfwp connects to the X server associated with this listen port and performs authorization checks against the server as well as against its own configurable access control list for requesting clients. If these checks fail, or if the requested server does not support the X Security Extension, the client connection is refused. Otherwise, the connection is accepted and all ensuing data between client and server is relayed by xfwp until the client terminates the connection or, in the case of an inactive client, until a configured timeout period is exceeded. Xfwp is designed to block while waiting for activity on its connections, thereby minimizing demand for system cycles.

If xfwp is run without a configuration file and thus no sitepolicy is defined, if xfwp is using an X server where xhost + has been run to turn off host-based authorization checks, when a client tries to connect to this X server via xfwp, the X server will deny the connection. If xfwp does not define a sitepolicy, host-based authorization must be turned on for clients to connect to an X server via the xfwp.

INTEROPERATION WITH IP PACKET-FILTERING ROUTERS

The whole purpose of the xfwp is to provide reliable control over access to Intranet X servers by clients originating outside the firewall. At the present time, such access control is typically achieved by firewall configurations incorporating IP packet-filtering routers. Frequently, the rules for such filters deny access to X server ports (range 6000 - 6xxx) for all Intranet host machines.

In order for xfwp to do its job, restrictions on access for ports 6001 - 6xxx must be removed from the rule-base of the IP packet-filtering router. [NOTE: xfwp only assigns ports in the range beginning with 6001; access to port 6000 on all Intranet hosts may continue to be denied.] This does not mean the Intranet firewall will be opened for indiscriminate entry by X clients. Instead, xfwp supports a fully configurable rule-based access control system, similar to that of the IP packet-filter router itself. Xfwp in effect adds another level of packet-filtering control which is fully configurable and applies specifically to X traffic. See section entitled CONFIGURATION FILE, below, for further details.

INSTALLATION, SETUP AND TROUBLESHOOTING

Xfwp is typically run as a background process on the Intranet firewall host. It can be launched using any of the command-line options described above. As noted above, xfwp works only in conjunction with proxy manager and the xfindproxy utility. It can also be configured to support a user-defined X server site security policy, in which the X server is required to indicate to xfwp whether or not it supports the particular policy. Consult the X server man pages for further information on these components. Xfwp diagnostics can be turned on by compiling with the -DDEBUG switch. Connection status can be recorded by using the -logfile and -loglevel command line options.

PERFORMANCE, LOAD BALANCING AND RESOURCE MANAGEMENT

Xfwp manages four different kinds of connections: proxy manager (PM) data, X client listen, X client data, and X server. The sysadmin employing xfwp must understand how the resources for each of these connection types are allocated and reclaimed by xfwp in order to optimize the availability of xfwp service.

Each connection-type has a default number of allocation slots and a default timeout. The number of allocation slots for PM connections and X server connections is configurable via command line options. Connection timeouts are also configurable via command line options. Each connection timeout represents the period the connection will be allowed to remain open in the absence of any activity on that connection.

Whenever there is activity on a connection, the time-to-close is automatically reset. The default distribution of total process connection slots across the four connection types, as well as the choice of default timeouts for the connection types, is governed by a number of assumptions embedded in the xfwp use model.

The default number of PM connections is 10 and the default duration for PM connections is 3,600 seconds (1 hour) for each connection after time of last activity. At start-up, xfwp listens for PM connection requests on any non-reserved port (default of 4444 if not specified on the xfwp command-line). The PM normally connects to xfwp only when a call is made to the PM with xfindproxy. Thereafter, the PM remains connected to xfwp, even after the messaging between them has been completed, for the default connection duration period. In some cases this may result in depletion of available PM connection slots. If the sysadmin expects connections to a single xfwp from many PM's, xfwp should be started using the -pdt command line option, with a timeout value reflecting the desired duration that inactive connections will be permitted to remain open.

Xfwp client listeners are set up by a call to xfindproxy and continue to listen for X client connection requests for a default duration of 86,400 seconds (24 hours) from the point of last activity. After this time they are automatically closed and their fd's recovered for future allocation. In addressing the question of how to choose some alternative timeout value which will guarantee the availability of client listen ports, sysadmins should take into consideration the expected delay between the time when the listener was allocated (using xfindproxy) and the time when a client actually attempts to connect to xfwp, as well the likelihood that client listeners will be re-used after the initial client data connection is closed.

Each client connection is allocated a default lifetime of 604,800 seconds (7 * 24 hours) from the point when it last saw activity. After this time it is automatically closed and its fd's recovered for future allocation. Because server connections are not actually established until a connection request from a remote X client arrives at one of the xfwp's client listen ports, the client data timeout applies both to client-xfwp connections as well as to xfwp-server connections. If the system administrator expects many client data connections through xfwp, an overriding of the default timeout should be considered.

CONFIGURATION FILE

The xfwp configuration file resides on the xfwp host machine and is used to determine whether X client data connection requests will be permitted or denied. The path to the file is specified at start-up time. If no configuration file is specified, all X client data connection requests routed through xfwp will be by default permitted, assuming that other X server authorization checks are successful. If a configuration file is supplied but none of its entries matches the connection request then the connection is by default denied.

If a line in the configuration file begins with the '#' character or a new-line character, the line is ignored and the evaluator will skip the line.

The configuration file supports two entirely independent authorization checks: one which is performed by xfwp itself, and a second which is the result of xfwp's querying the target X server. For the first of these, the configuration file employs a syntax and semantic similar to that of IP packet-filtering routers. It contains zero or more source-destination rules of the following form:

{permit | deny} <src> <src mask> [<dest> <dest mask> [<operator> <service>]]

permit/deny the keywords "permit" or "deny" indicate whether the rule will enable or disable access, respectively

src the IP address against the host who originated the connection request will be matched, expressed in IP format (x.x.x.x)

src mask a subnet mask, also in IP format, for further qualifying the source mask. Bits set in the mask indicate bits of the incoming address to be *ignored* when comparing to the specified src dest — the IP address against which the destination of the incoming connection request (i.e. the host IP of the X server to which the incoming client is attempting to connect) will be matched

dest mask — a subnet mask, also in IP format, for further qualifying the destination mask. Bits set in the mask indicate bits of the destination address to be *ignored* when comparing to the specified dest

operator — always ‘‘eq’’ (if the service field is not NULL)

service — one of the following three strings: ‘‘pm’’, ‘‘fp’’, or ‘‘cd’’, corresponding to proxy manager, xfindproxy, or client data, respectively

For the second type of authorization check, the configuration file contains zero or more site policy rules of the following form:

{require | disallow} sitepolicy <site_policy>

require — specifies that the X server *must* be configured with *at least one* of the corresponding site policies, else it must refuse the connection.

disallow — specifies that the X server *must not* be configured with *any* of the corresponding site policies, else it must refuse the connection.

sitepolicy — a required keyword

<site_policy> — specifies the policy string. The string may contain any combination of alphanumeric characters subject only to interpretation by the target X server

RULES FOR EVALUATING THE XFWP CONFIGURATION FILE ENTRIES

For the first type of configurable authorization checking, access can be permitted or denied for each connection type based upon source and, optionally, destination and service. Each file entry must at a minimum specify the keyword ‘‘permit’’ or ‘‘deny’’ and the two source fields. The destination and service fields can be used to provide finer-grained access control if desired.

The algorithm for rule-matching is as follows:

```
while (more entries to check)
{
 if ((<originator IP> AND (NOT <src mask>)) == src)
   [if ((<dest X server IP> AND (NOT <dest mask>)) == dest)]
     [if (service fields present and matching)]
       do either permit or deny connection depending on keyword
 else
   continue
}
if (no rule matches)
 deny connection
```

If a permit or deny rule does not specify a service and operation, then the rule applies to all services. If a configuration file is specified and it contains at least one valid deny or permit rule, then a host that is not explicitly permitted will be denied a connection.

Site policy configuration checking constitutes a separate (and X server only) authorization check on incoming connection requests. Any number of require or disallow rules may be specified, but all rules must be of the same type; that is, a single rule file cannot have both ‘‘require’’ and ‘‘disallow’’ keywords. The algorithm for this check is as follows:

```
if (X server recognizes any of the site policy strings)
 if (keyword == require)
   permit connection
 else
```

```
    deny connection
  else
   if (keyword == require)
    deny connection
   else
    permit connection
```

The site policy check is performed by xfwp only if the source-destination rules permit the connection.

EXAMPLES

```
# if and only if server supports one of these policies then authorize
# connections, but still subject to applicable rule matches
#
require sitepolicy policy1
require sitepolicy policy2
#
# deny pm connections originating on 8.7.6.5 [NOTE:  If pm service
# is explicitly qualified, line must include destination fields as
# shown.]
#
deny  8.7.6.5  0.0.0.0  0.0.0.0  255.255.255.255  eq  pm
#
# permit xfindproxy X server connects to anywhere [NOTE:  If
# fp service is explicitly qualified, line must include source fields
# as shown.]
#
permit  0.0.0.0  255.255.255.255   0.0.0.0  255.255.255.255  eq  fp
#
# permit all connection types originating from the 192.0.0.0
# IP domain only
#
permit  192.0.0.0   0.255.255.255
```

Care should be taken that source-destination rules are written in the correct order, as the first matching rule will be applied. In addition to parser syntax checking, a special command-line switch (-verify) has been provided to assist the sysadmin in determining which rule was actually matched.

BUGS

Xfwp should check server site policy and security extension before allocating a listen port.

SEE ALSO

xfindproxy (1), Proxy Management Protocol spec V1.0, proxymngr(1), Xserver(1)

AUTHOR

Reed Augliere, consulting to X Consortium, Inc.

NAME

xgamma - Alter a monitor's gamma correction for XFree86

SYNOPSIS

xgamma [-display *display*] [-screen *screen*] [-quiet] [-gamma f.f | [[-rgamma f.f] [-ggamma f.f] [-bgamma f.f]]]

DESCRIPTION

xgamma allows X users to query and alter the gamma correction of a monitor via the XFree86 X server video mode extension (XFree86-VidModeExtension).

OPTIONS

-display *display*
: This argument allows you to specify the server to connect to; see *X(7)*.

-screen *screen*
: When multiple displays are configured as a single logical display, this option allows you to select the screen you wish to change.

-quiet Silence the normal output of **xgamma**

-help Print out the 'Usage:' command syntax summary.

-gamma f.f
: The gamma correction can either be defined as a single value, or separately for the red, green and blue components. This argument specifies the gamma correction as a single value. If no value for the gamma correction is given **xgamma** returns the current gamma correction of the display.

-rgamma f.f
: This argument specifies the red component of the gamma correction.

-ggamma f.f
: This argument specifies the green component of the gamma correction.

-bgamma f.f
: This argument specifies the blue component of the gamma correction.

ENVIRONMENT

DISPLAY
: To get default host and display number.

BUGS

This client changes the internal values of the gamma correction for the Xserver. Whether or not these values are respected depends on the video drivers.

The gamma values are passed to the Xserver with 3 decimal places of accuracy.

SEE ALSO

xvidtune(1)

AUTHORS

Kaleb S. Keithley, X Consortium.
David Dawes, David Bateman

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(7)

BUGS

This program isn't really finished yet.
See *X(7)* for a full statement of rights and permissions.

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.

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.

−help Prints a usage message.

[+]*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.

NAMES

A complete name has the syntax ''family:name'' where the families are as follows:

inet	Internet host (IPv4)
inet6	Internet host (IPv6)
dnet	DECnet host
nis	Secure RPC network name
krb	Kerberos V5 principal
local	contains only one name, the empty string

The family is case insensitive. The format of the name varies with the family.

When Secure RPC is being used, the network independent netname (e.g., "nis:unix.*uid@domainname*") can be specified, or a local user can be specified with just the username and a trailing at-sign (e.g., "nis:pat@").

For backward compatibility with pre-R6 *xhost*, names that contain an at-sign (@) are assumed to be in the nis family. Otherwise they are assumed to be Internet addresses. If compiled to support IPv6, then all IPv4 and IPv6 addresses returned by getaddrinfo(3) are added to the access list in the appropriate inet or inet6 family.

DIAGNOSTICS

For each name added to the access control list, a line of the form "*name* being added to access control list" is printed. For each name removed from the access control list, a line of the form "*name* being removed from access control list" is printed.

FILES

/etc/X*.hosts

SEE ALSO

X(7), Xsecurity(7), Xserver(1), xdm(1), getaddrinfo(3)

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).

The X server stores network addresses, not host names. This is not really a bug. 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.

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.

Note, when using a *.xserverrc* script be sure to ‘‘exec’’ the real X server. Failing to do this can make the X server slow to start and exit. For example:

```
exec Xdisplaytype
```

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/X11R6/bin/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/X11R6/bin/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(7), *startx*(1), *Xserver*(1), *xterm*(1)

AUTHOR

Bob Scheifler, MIT Laboratory for Computer Science

NAME

xkbcomp − compile XKB keyboard description

SYNOPSIS

xkbcomp [option] source [destination]

DESCRIPTION

The *xkbcomp* keymap compiler converts a description of an XKB keymap into one of several output formats. The most common use for xkbcomp is to create a compiled keymap file (.xkm extension) which can be read directly by XKB-capable X servers or utilities. The keymap compiler can also produce C header files or XKB source files. The C header files produced by xkbcomp can be included by X servers or utilities that need a built-in default keymap. The XKB source files produced by *xkbcomp* are fully resolved and can be used to verify that the files which typically make up an XKB keymap are merged correctly or to create a single file which contains a complete description of the keymap.

The *source* may specify an X display, or an *.xkb* or *.xkm* file; unless explicitly specified, the format of *destination* depends on the format of the source. Compiling a *.xkb* (keymap source) file generates a *.xkm* (compiled keymap file) by default. If the source is a *.xkm* file or an X display, *xkbcomp* generates a keymap source file by default.

If the *destination* is an X display, the keymap for the display is updated with the compiled keymap.

The name of the *destination* is usually computed from the name of the source, with the extension replaced as appropriate. When compiling a single map from a file which contains several maps, *xkbcomp* constructs the destination file name by appending an appropriate extension to the name of the map to be used.

OPTIONS

−a Show all keyboard information, reporting implicit or derived information as a comment. Only affects *.xkb* format output.

−C Produce a C header file as output (.h extension).

−dflts Compute defaults for any missing components, such as key names.

−I*dir* Specifies top-level directories to be searched for files included by the keymap description. After all directories specified by −I options have been searched, the current directory and finally, the default xkb directory (usually /usr/X11R6/lib/X11/xkb) will be searched.

To prevent the current and default directories from being searched, use the −I option alone (i.e. without a directory), before any −I options that specify the directories you do want searched.

−l List maps that specify the *map* pattern in any files listed on the command line (not implemented yet).

−m *name*
Specifies a map to be compiled from an file with multiple entries.

−merge Merge the compiled information with the map from the server (not implemented yet).

−o *name* Specifies a name for the generated output file. The default is the name of the source file with an appropriate extension for the output format.

−opt *parts*
Specifies a list of optional parts. Compilation errors in any optional parts are not fatal. Parts may consist of any combination of the letters *c*, *g*,*k*,*s*,*t* which specify the compatibility map, geometry, keycodes, symbols and types, respectively.

-R*dir* Specifies the root directory for relative path names.

-synch Force synchronization for X requests.

−w *lvl* Controls the reporting of warnings during compilation. A warning level of 0 disables all warnings; a warning level of 10 enables them all.

−xkb Generate a source description of the keyboard as output (.xkb extension).

−xkm Generate a compiled keymap file as output (.xkm extension).

SEE ALSO

X(7)

COPYRIGHT

AUTHOR

Erik Fortune, Silicon Graphics

NAME

xkbevd − XKB event daemon

SYNOPSIS

xkbevd [options]

DESCRIPTION

This command is very raw and is therefore only partially implemented; we present it here as a rough prototype for developers, not as a general purpose tool for end users. Something like this might make a suitable replacement for xev; I'm not signing up, mind you, but it's an interesting idea.

The *xkbevd* event daemon listens for specified XKB events and executes requested commands if they occur. The configuration file consists of a list of event specification/action pairs and/or variable definitions.

An event specification consists of a short XKB event name followed by a string or identifier which serves as a qualifier in parentheses; empty parenthesis indicate no qualification and serve to specify the default command which is applied to events which do not match any of the other specifications. The interpretation of the qualifier depends on the type of the event: Bell events match using the name of the bell, message events match on the contents of the message string and slow key events accept any of *press*, *release*, *accept*, or *reject*. No other events are currently recognized.

An action consists of an optional keyword followed by an optional string argument. Currently, *xkbev* recognizes the actions: *none*, *ignore*, *echo*, *printEvent*, *sound*, and *shell*. If the action is not specified, the string is taken as the name of a sound file to be played unless it begins with an exclamation point, in which case it is taken as a shell command.

Variable definitions in the argument string are expanded with fields from the event in question before the argument string is passed to the action processor. The general syntax for a variable is either $*cP or $(str), where c is a single character and str is a string of arbitrary length. All parameters have both single-character and long names.*

The list of recognized parameters varies from event to event and is too long to list here right now. This is a developer release anyway, so you can be expected to look at the source code (evargs.c is of particular interest).

The *ignore*, *echo*, *printEvent*, *sound*,and *shell* actions do what you would expect commands named *ignore*, *echo*, *printEvent*, *sound*, and *shell* to do, except that the sound command has only been implemented and tested for SGI machines. It launches an external program right now, so it should be pretty easy to adapt, especially if you like audio cues that arrive about a half-second after you expect them.

The only currently recognized variables are *soundDirectory* and *soundCmd*. I'm sure you can figure out what they do.

OPTIONS

−help Prints a usage message that is far more up-to-date than anything in this man page.

−cfg *file* Specifies the configuration file to read. If no configuration file is specified, *xkbevd* looks for ˜/.xkb/xkbevd.cf and $(LIBDIR)/xkb/xkbevd.cf in that order.

−sc *cmd* Specifies the command used to play sounds.

−sd *directory*
Specifies a top-level directory for sound files.

−display *display*
Specifies the display to use. If not present, *xkbevd* uses $DISPLAY.

−bg Tells *xkbevd* to fork itself (and run in the background).

−synch Forces synchronization of all X requests. Slow.

−v Print more information, including debugging messages. Multiple specifications of *-v* cause more output, to a point.

SEE ALSO

X(7)

COPYRIGHT

AUTHOR

Erik Fortune, Silicon Graphics

NAME

xkbprint − print an XKB keyboard description

SYNOPSIS

xkbprint [options] source [output_file]

DESCRIPTION

The *xkbprint* comman generates a printable or encapsulated PostScript description of the XKB keyboard description specified by *source*. The *source* can be any compiled keymap (.xkm) file that includes a geometry description or an X display specification. If an *output_file* is specified, xkbprint writes to it. If no output file is specified, xkbprint creates replaces the extension of the source file with *.ps* or *.eps* depending on the requested format. If the source is a non-local X display (e.g.:0), xkbprint appends the appropriate prefix to the display specification, replacing the colon with a dash. For a local display, xkprint uses server-*n* where *n* is the number of the display.

OPTIONS

−?, -help
: Prints a usage message.

−color
: Print using the colors specified in the geometry file; by default, xkbprint prints a black-and-white image of the keyboard.

−dflts
: Attempt to compute default names for any missing components, such as keys.

−diffs
: Show symbols only where they are explicitly bound.

−eps
: Generate an encapsulated PostScript file.

−fit
: Fit the keyboard image on the page (default).

−full
: Print the keyboard at full size.

−grid *res*
: Print a grid with *res*mm resolution over the keyboard.

−if *fontName*
: Specifies an internal PostScript type 1 font to dump to the specified output file or to *fontName*.pfa, if no output file is specified. No keyboard description is printed if an internal font is dumped.

−label *type*
: Specifies the labels to be printed on keys; legal types are: *none*, *name*,*code*,*symbols*.

−lc *<locale>*
: Specifies a locale in which KeySyms should be resolved.

−level1
: Generate level 1 PostScript.

−level2
: Generate level 2 PostScript.

−lg *group*
: Print symbols in keyboard groups starting from *group*.

−ll *level*
: Print symbols starting from shift level *level*.

−mono
: Generate black-and-white image of keyboard (default).

−n *num*
: Print *num* copies.

−nkg *num*
: Print the symbols in *num* keyboard groups.

−npk *num*
: Number of keyboard images to print on each page; for EPS files, this specifies the total number of keyboard images to print.

−o *file*
: Write output to *file*.

–R*directory*
Use *directory* as the root directory; all path names are interpreted relative to *directory*.

-pict *which*
Controls use of pictographs instead of keysym names where available. *which* can be any of *all*, *none*, or *common*(default).

-synch Forces synchronization for X requests.

-w *level* Sets warning level (0 for no warning, 10 for all warnings).

SEE ALSO

X(7),xkbcomp(1)

COPYRIGHT

AUTHOR

Erik Fortune, Silicon Graphics

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.

CAVEATS

This command does not provide any warranty that the application whose connection to the X server is closed will abort nicely, or even abort at all. All this command does is to close the connection to the X server. Many existing applications do indeed abort when their connection to the X server is closed, but some can choose to continue.

XDEFAULTS

Button Specifies a specific pointer button number or the word "any" to use when selecting windows.

SEE ALSO

X(7), xwininfo(1), XKillClient and XGetPointerMapping in the Xlib Programmers Manual, KillClient in the X Protocol Specification

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*] [-remote *host*] [-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(7)*). 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.

−remote *host*
This option tells *xload* to display the load of *host* instead of *localhost*. *Xload* gets the information from the *rwhod* database and consequently requires *rwhod* to be executing both on *localhost* and *host*.

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/X11R6/lib/X11/app-defaults/XLoad
specifies required resources

SEE ALSO

X(7), 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). DG/UX support by Takis Psarogiannakopoulos (XFree86 Project).

NAME

xlogo - X Window System logo

SYNOPSIS

xlogo [-*toolkitoption* ...]

DESCRIPTION

The *xlogo* program displays the X Window System logo.

OPTIONS

Xlogo accepts all of the standard X Toolkit command line options, as well as the following:

−render This option indicates that the logo should be drawn with anti-aliased edges using the RENDER extension.

−sharp If -render is also specified, this forces the edges to be rendered in sharp mode, (ie. 1-bit alpha channel).

−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 *XtDefaultBackground*.

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/X11R6/lib/X11/app-defaults/XLogo
specifies required resources

SEE ALSO

X(7), xrdb(1)

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(7), Xserver(1), xprop(1)

ENVIRONMENT
DISPLAY
to get the default host and display to use.

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 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 List in long format, giving the window name, icon name, and class hints in addition to the machine name and command string shown in the default format.

–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(7), xwininfo(1), xprop(1)

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 Lists some attributes of the font on one line in addition to its name.

−ll Lists font properties in addition to **−l** output.

−lll Lists character metrics in addition to **−ll** output.

−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).

−fn *pattern*
: This option specifies the font name pattern to match.

SEE ALSO

X(7), Xserver(1), xset(1), xfd(1), *X Logical Font Description Conventions*

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.

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 "ˆC" 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.

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*.

BSD AND LINUX SYSTEMS

In newer BSD and Linux systems, *Xman* will search for a file named */etc/man.conf* which will contain the list of directories containing manual pages. See *man.conf*(5) for a complete description of the file format.

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/X11R6/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(7), *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. See *X(7)* for a full statement of rights and permissions.

AUTHORS

Chris Peterson, MIT X Consortium from the V10 version written by Barry Shein formerly of Boston University. Bug fixes and Linux support by Carlos A M dos Santos, The XFree86 Project.

NAME

xmessage − display a message or query in a window (X-based /bin/echo)

SYNOPSIS

xmessage [**−buttons** *label1*[**:***value1*]**,***label2*[**:***value2*]**,** . . .] [*options*] **−file** *filename*
xmessage [**−buttons** *label1*[**:***value1*]**,***label2*[**:***value2*]**,** . . .] [*options*] *message* . . .

DESCRIPTION

The *xmessage* program displays a window containing a message from the command line, a file, or standard input. Along the lower edge of the message is row of buttons; clicking the left mouse button on any of these buttons will cause *xmessage* to exit. Which button was pressed is returned in the exit status and, optionally, by writing the label of the button to standard output.

The program is typically used by shell scripts to display information to the user or to ask the user to make a choice.

Unless a size is specified, *xmessage* sizes itself to fit the message, up to a maximum size. If the message is too big for the window, *xmessage* will display scroll bars.

OPTIONS

These are the command line options that *xmessage* understands.

−buttons *button,button,. . .*
This option will cause *xmessage* to create one button for each comma-separated *button* argument. The corresponding resource is **buttons**. Each *button* consists of a label optionally followed by a colon and an exit value. The label is the name of the Command button widget created and will be the default text displayed to the user. Since this is the name of the widget it may be used to change any of the resources associated with that button. The exit value will be returned by *xmessage* if that button is selected. The default exit value is 100 plus the button number. Buttons are numbered from the left starting with one. The default string if no **−buttons** option is given is **okay:0**.

−default *label*
Defines the button with a matching *label* to be the default. If not specified there is no default. The corresponding resource is **defaultButton**. Pressing Return anywhere in the *xmessage* window will activate the default button. The default button has a wider border than the others.

−file *filename*
File to display. The corresponding resource is **file**. A *filename* of '**−**' reads from standard input. If this option is not supplied, *xmessage* will display all non-option arguments in the style of *echo*. Either **−file** or a message on the command line should be provided, but not both.

−print This will cause the program to write the label of the button pressed to standard output. Equivalent to setting the **printValue** resource to TRUE. This is one way to get feedback as to which button was pressed.

−center Pop up the window at the center of the screen. Equivalent to setting the **center** resource to TRUE.

−nearmouse
Pop up the window near the mouse cursor. Equivalent to setting the **nearMouse** resource to TRUE.

−timeout *secs*
Exit with status 0 after *secs* seconds if the user has not clicked on a button yet. The corresponding resource is **timeout**.

WIDGET HIERARCHY

Knowing the name and position in the hierarchy of each widget is useful when specifying resources for them. In the following chart, the class and name of each widget is given.

Xmessage (xmessage)
    Form form

Text message
Command (label1)
Command (label2)
.
.
.

RESOURCES

The program has a few top-level application resources that allow customizations that are specific to *xmessage*.

file A String specifying the file to display.

buttons A String specifying the buttons to display. See the **−buttons** command-line option.

defaultButton
A String specifying a default button by label.

printValue
A Boolean value specifying whether the label of the button pressed to exit the program is written to standard output. The default is FALSE.

center A Boolean value specifying whether to pop up the window at the center of the screen. The default is FALSE.

nearMouse
A Boolean value specifying whether to pop up the window near the mouse cursor. The default is FALSE.

timeout The number of seconds after which to exit with status 0. The default is 0, which means never time out.

maxHeight (class **Maximum)**
The maximum height of the text part of the window in pixels, used if no size was specified in the geometry. The default is 0, which means use 70% of the height of the screen.

maxWidth (class **Maximum)**
The maximum width of the text part of the window in pixels, used if no size was specified in the geometry. The default is 0, which means use 70% of the width of the screen.

ACTIONS

exit(*value***)**
exit immediately with an exit status of *value* (default 0). This action can be used with translations to provide alternate ways of exiting *xmessage*.

default-exit()
exit immediately with the exit status specified by the default button. If there is no default button, this action has no effect.

EXIT STATUS

If it detects an error, *xmessage* returns 1, so this value should not be used with a button.

SEE ALSO

X(7), *echo*(1), *cat*(1)

AUTHORS

Chris Peterson, MIT Project Athena
Stephen Gildea, X Consortium

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(7)*.

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 pull-down 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 **XmhMarkDelete()**.

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.

When *xmh* is run under a Release 6 session manager it will prompt the user for confirmation during a checkpoint operation. The dialog box asks whether any current changes should be committed (saved) during the checkpoint. Responding ‘‘Yes’’ will have the same effect as pressing the ‘‘Commit Changes’’ or ‘‘Save Message’’ buttons in the respective folder and view windows. Responding ‘‘No’’ will cause the checkpoint to continue successfully to completion without actually saving any pending changes. If the session manager disallows user interaction during the checkpoint a ‘‘Yes’’ response is assumed; i.e. all changes will be committed during the checkpoint.

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 shows the program name, vendor, and release.

blockEventsOnBusy
Whether to disallow user input and show a busy cursor while *xmh* is busy processing a command. If false, the user can 'mouse ahead' and type ahead; if true, user input is discarded when processing lengthy *mh* commands. The 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 occurrence 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 forwarding 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 commands, 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 messages, 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(7), 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.
- Should save a temporary checkpoint file rather than requiring changes to be committed in the non-shutdown case.

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 X Consortium build tree.

If working inside the X Consortium 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 and pointer button mappings 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). Up to eight keysyms may be attached to a key, however the last four are not used in any major X server implementation. The first keysym is used when no modifier key is pressed in conjunction with this key, the second with Shift, the third when the Mode_switch key is used with this key and the fourth when both the Mode_switch and Shift keys are used.

keycode any = *KEYSYMNAME ...*

If no existing key has the specified list of keysyms assigned to it, a spare key on the keyboard is selected and the keysyms are assigned to it. The list of keysyms may be specified in decimal, hex or octal.

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 *__projectroot__/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 the 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 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"
```

Similarly, some keyboards have an Alt key but no Meta key. In that case the following may be useful:

```
% xmodmap -e "keysym Alt_L = Meta_L Alt_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 ^?" | 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(7), 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.

AUTHOR

Jim Fulton, MIT X Consortium, rewritten from an earlier version by David Rosenthal of Sun Microsystems.

NAME

Xnest − a nested X server

SYNOPSIS

Xnest [-options]

DESCRIPTION

Xnest is a client and a server. *Xnest* is a client of the real server which manages windows and graphics requests on its behalf. *Xnest* is a server to its own clients. *Xnest* manages windows and graphics requests on their behalf. To these clients *Xnest* appears to be a conventional server.

OPTIONS

Xnest supports all standard options of the sample server implementation. For more details, please see the manual page on your system for *Xserver*. The following additional arguments are supported as well.

−display *string*
: This option specifies the display name of the real server that *Xnest* should try to connect with. If it is not provided on the command line *Xnest* will read the *DISPLAY* environment variable in order to find out the same information.

−sync
: This option tells *Xnest* to synchronize its window and graphics operations with the real server. This is a useful option for debugging, but it will slow down the performance considerably. It should not be used unless absolutely necessary.

−full
: This option tells *Xnest* to utilize full regeneration of real server objects and reopen a new connection to the real server each time the nested server regenerates. The sample server implementation regenerates all objects in the server when the last client of this server terminates. When this happens, *Xnest* by default maintains the same top level window and the same real server connection in each new generation. If the user selects full regeneration, even the top level window and the connection to the real server will be regenerated for each server generation.

−class *string*
: This option specifies the default visual class of the nested server. It is similar to the *-cc* option from the set of standard options except that it will accept a string rather than a number for the visual class specification. The string must be one of the following six values: *StaticGray*, *GrayScale*, *StaticColor*, *PseudoColor*, *TrueColor*, or *DirectColor*. If both, *-class* and *-cc* options are specified, the last instance of either option assumes precedence. The class of the default visual of the nested server need not be the same as the class of the default visual of the real server; although, it has to be supported by the real server. See *xdpyinfo* for a list of supported visual classes on the real server before starting *Xnest*. If the user chooses a static class, all the colors in the default colormap will be preallocated. If the user chooses a dynamic class, colors in the default colormap will be available to individual clients for allocation.

−depth *int*
: This option specifies the default visual depth of the nested server. The depth of the default visual of the nested server need not be the same as the depth of the default visual of the real server; although, it has to be supported by the real server. See *xdpyinfo* for a list of supported visual depths on the real server before starting *Xnest*.

−sss
: This option tells *Xnest* to use the software screen saver. By default *Xnest* will use the screen saver that corresponds to the hardware screen saver in the real server. Of course, even this screen saver is software generated since *Xnest* does not control any actual hardware. However, it is treated as a hardware screen saver within the sample server code.

−geometry *WxH+X+Y*
: This option specifies geometry parameters for the top level *Xnest* windows. These windows corresponds to the root windows of the nested server. The width and height specified with this option

will be the maximum width and height of each top level *Xnest* window. *Xnest* will allow the user to make any top level window smaller, but it will not actually change the size of the nested server root window. As of yet, there is no mechanism within the sample server implementation to change the size of the root window after screen initialization. In order to do so, one would probably need to extend the X protocol. Therefore, it is not likely that this will be available any time soon. If this option is not specified *Xnest* will choose width and height to be 3/4 of the dimensions of the root window of the real server.

−bw *int*
This option specifies the border width of the top level *Xnest* window. The integer parameter must be a positive number. The default border width is 1.

−name *string*
This option specifies the name of the top level *Xnest* window. The default value is the program name.

−scrns *int*
This option specifies the number of screens to create in the nested server. For each screen, *Xnest* will create a separate top level window. Each screen is referenced by the number after the dot in the client display name specification. For example, *xterm -display :1.1* will open an *xterm* client in the nested server with the display number *:1* on the second screen. The number of screens is limited by the hard coded constant in the server sample code which is usually 3.

−install
This option tells *Xnest* to do its own colormap installation by bypassing the real window manager. For it to work properly the user will probably have to temporarily quit the real window manager. By default *Xnest* will keep the nested client window whose colormap should be installed in the real server in the *WM_COLORMAP_WINDOWS* property of the top level *Xnest* window. If this colormap is of the same visual type as the root window of the nested server, *Xnest* will associate this colormap with the top level *Xnest* window as well. Since this does not have to be the case, window managers should look primarily at the *WM_COLORMAP_WINDOWS* property rather than the colormap associated with the top level *Xnest* window. Unfortunately, window managers are not very good at doing that yet so this option might come in handy.

−parent *window_id*
This option tells *Xnest* to use the *window_id* as the root window instead of creating a window. This option is used by the xrx xnestplugin.

USAGE

Starting up *Xnest* is as simple as starting up *xclock* from a terminal emulator. If a user wishes to run *Xnest* on the same workstation as the real server, it is important that the nested server is given its own listening socket address. Therefore, if there is a server already running on the user's workstation, *Xnest* will have to be started up with a new display number. Since there is usually no more than one server running on a workstation, specifying *Xnest :1* on the command line will be sufficient for most users. For each server running on the workstation the display number needs to be incremented by one. Thus, if you wish to start another *Xnest*, you will need to type *Xnest :2* on the command line.

To run clients in the nested server each client needs to be given the same display number as the nested server. For example, *xterm -display :1* will start up an *xterm* in the first nested server and *xterm -display :2* will start an *xterm* in the second nested server from the example above. Additional clients can be started from these *xterm*s in each nested server.

XNEST AS A CLIENT

Xnest behaves and looks to the real server and other real clients as another real client. It is a rather demanding client, however, since almost any window or graphics request from a nested client will result in a window or graphics request from *Xnest* to the real server. Therefore, it is desirable that *Xnest* and the real server are on a local network, or even better, on the same machine. As of now, *Xnest* assumes that the real server supports the shape extension. There is no way to turn off this assumption dynamically. *Xnest* can be compiled without the shape extension built in, and in that case the real server need not support it. The dynamic shape extension selection support should be considered in further development of *Xnest*.

Since *Xnest* need not use the same default visual as the the real server, the top level window of the *Xnest* client always has its own colormap. This implies that other windows' colors will not be displayed properly while the keyboard or pointer focus is in the *Xnest* window, unless the real server has support for more than one installed colormap at any time. The colormap associated with the top window of the *Xnest* client need not be the appropriate colormap that the nested server wants installed in the real server. In the case that a nested client attempts to install a colormap of a different visual from the default visual of the nested server, *Xnest* will put the top window of this nested client and all other top windows of the nested clients that use the same colormap into the *WM_COLORMAP_WINDOWS* property of the top level *Xnest* window on the real server. Thus, it is important that the real window manager that manages the *Xnest* top level window looks at the *WM_COLORMAP_WINDOWS* property rather than the colormap associated with the top level *Xnest* window. Since most window managers appear to not implement this convention properly as of yet, *Xnest* can optionally do direct installation of colormaps into the real server bypassing the real window manager. If the user chooses this option, it is usually necessary to temporarily disable the real window manager since it will interfere with the *Xnest* scheme of colormap installation.

Keyboard and pointer control procedures of the nested server change the keyboard and pointer control parameters of the real server. Therefore, after *Xnest* is started up, it will change the keyboard and pointer controls of the real server to its own internal defaults. Perhaps there should be a command line option to tell *Xnest* to inherit the keyboard and pointer control parameters from the real server rather than imposing its own. This is a future consideration.

XNEST AS A SERVER

Xnest as a server looks exactly like a real server to its own clients. For the clients there is no way of telling if they are running on a real or a nested server.

As already mentioned, *Xnest* is a very user friendly server when it comes to customization. *Xnest* will pick up a number of command line arguments that can configure its default visual class and depth, number of screens, etc. In the future, *Xnest* should read a customization input file to provide even greater freedom and simplicity in selecting the desired layout. Unfortunately, there is no support for backing store and save under as of yet, but this should also be considered in the future development of *Xnest*.

The only apparent intricacy from the users' perspective about using *Xnest* as a server is the selection of fonts. *Xnest* manages fonts by loading them locally and then passing the font name to the real server and asking it to load that font remotely. This approach avoids the overload of sending the glyph bits across the network for every text operation, although it is really a bug. The proper implementation of fonts should be moved into the *os* layer. The consequence of this approach is that the user will have to worry about two different font paths - a local one for the nested server and a remote one for the real server - since *Xnest* does not propagate its font path to the real server. The reason for this is because real and nested servers need not run on the same file system which makes the two font paths mutually incompatible. Thus, if there is a font in the local font path of the nested server, there is no guarantee that this font exists in the remote font path of the real server. *Xlsfonts* client, if run on the nested server will list fonts in the local font path and if run on the real server will list fonts in the remote font path. Before a font can be successfully opened by the nested server it has to exist in local and remote font paths. It is the users' responsibility to make sure that this is the case.

BUGS

Won't run well on servers supporting different visual depths. Still crashes randomly. Probably has some memory leaks.

AUTHOR

Davor Matic, MIT X Consortium

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, remsh, or rcmd. 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 log in without a password, 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/remsh/rcmd 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, 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

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*] [-set *property-name value*] [-spy] [-f *atom format* [*dformat*]]* [*format* [*dformat*] *atom*]*

SUMMARY

The *xprop* 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(7)*.

-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.

-set *property-name value*
Specifies the name of a property and a property value, to be set on 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.

t This field and the next ones until either a 0 or the end of the property represent an internationalized text string. This format character is only usable with a field size of 8. The string is assumed to be in an ICCCM compliant encoding and is converted to the current locale encoding before being output.

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(7), xwininfo(1)

AUTHOR

Mark Lillibridge, MIT Project Athena

NAME

xrandr − primitive command line interface to RandR extension

SYNOPSIS

xrandr [-help] [-display *display*] [-o *orientation*] [-q] [-v] [-s *size*] [-x] [-y] [--screen *snum*] [--verbose]

DESCRIPTION

Xrandr is used to set the screen size, orientation and/or reflection. The *-s* option is a small integer index used to specify which size the screen should be set to. To find out what sizes are available, use the *-q* option, which reports the sizes available, the current rotation, and the possible rotations and reflections. The default size is the first size specified in the list. The *-o* option is used to specify the orientation of the screen, and can be one of *"normal inverted left right 0 1 2 3"*.

The *-x* option instructs the server to reflect the screen on the X axis. The *-y* option instructs the server to reflect the screen on the Y axis. Reflection is applied after rotation.

The *-help* option prints out a usage summary. The *--verbose* option tells you what xrandr is doing, selects for events, and tells you when events are received to enable debugging.

SEE ALSO

Xrandr(3)

AUTHORS

Keith Packard, XFree86 Core Team and Cambridge Research Laboratory, HP Labs, HP. and Jim Gettys, Cambridge Research Laboratory, HP Labs, HP.

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:

SERVERHOST=*hostname*
: the hostname portion of the display to which you are connected.

SRVR_*name*
: the SERVERHOST hostname string turned into a legal identifier. For example, "my-dpy.lcs.mit.edu" becomes SRVR_my_dpy_lcs_mit_edu.

HOST=*hostname*
: the same as **SERVERHOST**.

DISPLAY_NUM=*num*
: the number of the display on the server host.

CLIENTHOST=*hostname*
: the name of the host on which *xrdb* is running.

CLNT_*name*
: the CLIENTHOST hostname string turned into a legal identifier. For example, "expo.lcs.mit.edu" becomes CLNT_expo_lcs_mit_edu.

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 literal specifying the vendor of the server.

VNDR_*name*
: the VENDOR name string turned into a legal identifier. For example, "MIT X Consortium" becomes VNDR_MIT_X_Consortium.

EXT_*name*
A symbol is defined for each protocol extension supported by the server. Each extension string name is turned into a legal identifier. For example, "X3D-PEX" becomes EXT_X3D_PEX.

NUM_SCREENS=*num*
the total number of screens.

SCREEN_NUM=*num*
the number of the current screen (from zero).

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.

CLASS_*visualclass***=***visualid*
the visual class of the root window in a form you can *#ifdef* on. The value is the numeric id of the visual.

COLOR
defined only if CLASS is one of StaticColor, PseudoColor, TrueColor, or DirectColor.

CLASS_*visualclass_depth***=***num*
A symbol is defined for each visual supported for the screen. The symbol includes the class of the visual and its depth; the value is the numeric id of the visual. (If more than one visual has the same class and depth, the numeric id of the first one reported by the server is used.)

HEIGHT=*num*
the height of the root window in pixels.

WIDTH=*num*
the width of the root window in pixels.

PLANES=*num*
the number of bit planes (the depth) of the root window.

X_RESOLUTION=*num*
the x resolution of the screen in pixels per meter.

Y_RESOLUTION=*num*
the y resolution of the screen in pixels per meter.

SRVR_*name*, CLNT_*name*, VNDR_*name*, and EXT_*name* identifiers are formed by changing all characters other than letters and digits into underscores (_).

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(7)*. 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*, *-override* 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*, *-override* and *-merge*, the input file is processed for each screen.

–n This option indicates that changes to the specified properties (when used with *-load*, *-override* 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.

–override
This option indicates that the input should be added to, instead of replacing, the current contents of the specified properties. New entries override previous entries.

–merge This option indicates that the input should be merged and lexicographically sorted with, instead of replacing, the current contents of the specified properties.

–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

−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(7), 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.

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(7)*.

−display *display*
This argument allows you to specify the server and screen to refresh; see *X(7)*.

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(7)

BUGS

It should have just one default type for the background.

AUTHORS

Jim Gettys, Digital Equipment Corp., MIT Project Athena

NAME

xrx - RX helper program

SYNOPSIS

xrx [−*toolkitoption* ...] *filename*

DESCRIPTION

The helper program may be used with any Web browser to interpret documents in the RX MIME type format and start remote applications.

xrx reads in the RX document specified by its *filename*, from which it gets the list of services the application wants to use. Based on this information, **xrx** sets the various requested services, including creating authorization keys if your X server supports the SECURITY extension. It then passes the relevant data, such as the X display name, to the application through an HTTP GET request of the associated CGI script. The Web server then executes the CGI script to start the application. The client runs on the web server host connected to your X server.

INSTALLATION

You need to configure your web browser to use **xrx** for RX documents. Generally the following line in your $HOME/.mailcap is enough:

```
application/x-rx; xrx %s
```

However, you may need to refer to your web browser's documentation for exact instructions on configuring helper applications.

Once correctly configured, your browser will activate the helper program whenever you retrieve any document of the MIME type *application/x-rx*.

OPTIONS

The *xrx* helper program accepts all of the standard X Toolkit command line options such as:

−xrm *resourcestring*

This option specifies a resource string to be used. There may be several instances of this option on the command line.

RESOURCES

The application class name of the *xrx* program is Xrx and it understands the following application resource names and classes:

xrxHasFirewallProxy (class **XrxHasFirewallProxy)**

Specifies whether an X server firewall proxy (see xfwp) is running and should be used. Default is ''False.''

xrxInternalWebServers (class **XrxInternalWebServers)**

The web servers for which the X server firewall proxy should not be used (only relevant when **xrxHasFirewallProxy** is ''True''). Its value is a comma separated list of mask/value pairs to be used to filter internal web servers, based on their address. The mask part specifies which segments of the address are to be considered and the value part specifies what the result should match. For instance the following list:

```
255.255.255.0/198.112.45.0, 255.255.255.0/198.112.46.0
```

matches the address sets: 198.112.45.* and 198.112.46.*. More precisely, the test is (address & mask) == value.

xrxFastWebServers (class **XrxFastWebServers)**

The web servers for which LBX should not be used. The resource value is a list of address mask/value pairs, as previously described.

xrxTrustedWebServers (class **XrxTrustedWebServers)**

The web servers from which remote applications should be run as trusted clients. The default is to run remote applications as untrusted clients. The resource value is a list of address mask/value pairs, as previously described.

ENVIRONMENT

The *xrx* helper program uses the standard X environment variables such as ‘‘DISPLAY’’ to get the default X server host and display number. If the RX document requests X-UI-LBX service and the default X server does not advertise the LBX extension, *xrx* will look for the environment variable ‘‘XREALDISPLAY’’ to get a second address for your X server and look for the LBX extension there. When running your browser through *lbxproxy* you will need to set XREALDISPLAY to the actual address of your server if you wish remote applications to be able to use LBX across the Internet.

If the RX document requests XPRINT service, *xrx* looks for the variable ‘‘XPRINTER’’ to get the printer name and X Print server address to use. If the server address is not specified as part of XPRINTER, *xrx* uses the first one specified through the variable ‘‘XPSERVERLIST’’ when it is set. When it is not *xrx* then tries to use the video server as the print server. If the printer name is not specified via XPRINTER, *xrx* looks for it in the variables ‘‘PDPRINTER’’, then ‘‘LPDEST’’, and finally ‘‘PRINTER’’,

Finally, if you are using a firewall proxy, *xrx* will look for ‘‘PROXY_MANAGER’’ to get the address of your proxy manager (see proxymngr). When not specified it will use ":6500" as the default.

KNOWN BUG

When an authorization key is created for a remote application to use the X Print service, the helper program has to create the key with an infinite timeout since nobody knows when the application will actually connect to the X Print server. Therefore, in this case, the helper program stays around to revoke the key when the application goes away (that is when its video key expires). However, if the helper program dies unexpectedly the print authorization key will never get revoked.

SEE ALSO

libxrx (1), xfwp (1), lbxproxy (1), proxymngr (1), The RX Document specification

AUTHOR

Arnaud Le Hors, X Consortium

NAME

Xserver − X Window System display server

SYNOPSIS

X [option ...]

DESCRIPTION

X is the generic name for the X Window System display 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 X server is usually started from the X Display Manager program *xdm*(1) or a similar display manager program. 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.

Installations that run more than one window system may need to use the *xinit*(1) utility instead of a display manager. 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 a display manager, or build other interfaces for novice users.

The X server may also be started directly by the user, though this method is usually reserved for testing and is not recommended for normal operation. On some platforms, the user must have special permission to start the X server, often because access to certain devices (e.g. */dev/mouse*) is restricted.

When the X server starts up, it typically takes over the display. If you are running on a workstation whose console is the display, you may not be able to log into the console while the server is running.

OPTIONS

Many X servers have device-specific command line options. See the manual pages for the individual servers for more details; a list of server-specific manual pages is provided in the SEE ALSO section below.

All of the X servers accept the command line options described below. Some X servers may have alternative ways of providing the parameters described here, but the values provided via the command line options should override values specified via other mechanisms.

:*displaynumber*
: The X server runs as the given *displaynumber*, which by default is 0. If multiple X servers are to run simultaneously on a host, each must have a unique display number. See the DISPLAY NAMES section of the *X*(7) manual page to learn how to specify which display number clients should try to use.

−a *number*
: sets pointer acceleration (i.e. the ratio of how much is reported to how much the user actually moved the pointer).

−ac
: disables host-based access control mechanisms. Enables access by any host, and permits any host to modify the access control list. Use with extreme caution. This option exists primarily for running test suites remotely.

−audit *level*
: sets the audit trail level. The default level is 1, meaning only connection rejections are reported. Level 2 additionally reports all successful connections and disconnects. Level 4 enables messages from the SECURITY extension, if present, including generation and revocation of authorizations and violations of the security policy. Level 0 turns off the audit trail. Audit lines are sent as standard error output.

−auth *authorization-file*
: specifies a file which contains a collection of authorization records used to authenticate access. See also the *xdm*(1) and *Xsecurity*(7) 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.

−br sets the default root window to solid black instead of the standard root weave pattern.

−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. The default is */usr/X11R6/lib/X11/rgb*.

−core causes the server to generate a core dump on fatal errors.

−deferglyphs *whichfonts*
specifies the types of fonts for which the server should attempt to use deferred glyph loading. *whichfonts* can be all (all fonts), none (no fonts), or 16 (16 bit fonts only).

−dpi *resolution*
sets the resolution for all screens, in dots per inch. To be used when the server cannot determine the screen size(s) from the hardware.

dpms enables DPMS (display power management services), where supported. The default state is platform and configuration specific.

−dpms disables DPMS (display power management services). The default state is platform and configuration specific.

−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. See the FONTS section of this manual page for more information and the default list.

−help prints a usage message.

−I causes all remaining command line arguments to be ignored.

−maxbigreqsize *size*
sets the maxmium big request to *size* MB.

−nolisten *trans-type*
disables a transport type. For example, TCP/IP connections can be disabled with **−nolisten tcp**. This option may be issued multiple times to disable listening to different transport types.

−noreset
prevents a server reset when the last client connection is closed. This overrides a previous **−terminate** command line option.

−p *minutes*
sets screen-saver pattern cycle time in minutes.

−pn permits the server to continue running if it fails to establish all of its well-known sockets (connection points for clients), but establishes at least one. This option is set by default.

−nopn causes the server to exit if it fails to establish all of its well-known sockets (connection points for clients).

−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).

−terminate
causes the server to terminate at server reset, instead of continuing to run. This overrides a previous **−noreset** command line option.

−to *seconds*
sets default connection timeout in seconds.

−tst disables all testing extensions (e.g., XTEST, XTrap, XTestExtension1, RECORD).

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. This is a backdoor way of getting backing-store to apply to all windows. Although all mapped windows will have backing store, the backing store attribute value reported by the server for a window will be the last value established by a client. If it has never been set by a client, the server will report the default value, NotUseful. This behavior is required by the X protocol, which allows the server to exceed the client's backing store expectations but does not provide a way to tell the client that it is doing so.

−x *extension*
loads the specified extension at init. This is a no-op for most implementations.

[+-]xinerama
enables(+) or disables(-) the XINERAMA extension. The default state is platform and configuration specific.

SERVER DEPENDENT OPTIONS

Some X servers accept the following options:

−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.

−lf *files* sets the number-of-open-files limit of the server to the specified number. A value of zero makes the limit as large as possible. The default value of −1 leaves the limit unchanged.

−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.

−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.

−render **default|mono|gray|color** sets the color allocation policy that will be used by the render extension.

default selects the default policy defined for the display depth of the X server.

mono don't use any color cell.

gray use a gray map of 13 color cells for the X render extension.

color use a color cube of at most 4*4*4 colors (that is 64 color cells).

−dumbSched
disables smart scheduling on platforms that support the smart scheduler.

−schedInterval *interval*
sets the smart scheduler's scheduling interval to *interval* milliseconds.

XDMCP OPTIONS

X servers that support XDMCP have the following options. See the *X Display Manager Control Protocol* specification for more information.

−query *hostname*
enables XDMCP and sends Query packets to the specified *hostname*.

−broadcast
enable XDMCP and broadcasts BroadcastQuery packets to the network. The first responding display manager will be chosen for the session.

−multicast [*address* **[***hop count***]]**
Enable XDMCP and multicast BroadcastQuery packets to the network. The first responding display manager is chosen for the session. If an address is specified, the multicast is sent to that address. If no address is specified, the multicast is sent to the default XDMCP IPv6 multicast group. If a hop count is specified, it is used as the maximum hop count for the multicast. If no hop count is specified, the multicast is set to a maximum of 1 hop, to prevent the multicast from being routed beyond the local network.

−indirect *hostname*
enables XDMCP and send IndirectQuery packets to the specified *hostname*.

−port *port-number*
uses the specified *port-number* for XDMCP packets, instead of the default. This option must be specified before any −query, −broadcast, −multicast, or −indirect options.

−from *local-address*
specifies the local address to connect from (useful if the connecting host has multiple network interfaces). The *local-address* may be expressed in any form acceptable to the host platform's *gethostbyname*(3) implementation.

−once causes the server to terminate (rather than reset) when the XDMCP session ends.

−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.

XKEYBOARD OPTIONS

X servers that support the XKEYBOARD (a.k.a. "XKB") extension accept the following options. All layout files specified on the command line must be located in the XKB base directory or a subdirectory, and specified as the relative path from the XKB base directory. The default XKB base directory is */usr/X11R6/lib/X11/xkb*.

[+-]kb enables(+) or disables(-) the XKEYBOARD extension.

[+-]accessx [*timeout* [*timeout_mask* [*feedback* [*options_mask*]]]]
enables(+) or disables(-) AccessX key sequences.

–xkbdir *directory*
base directory for keyboard layout files. This option is not available for setuid X servers (i.e., when the X server's real and effective uids are different).

–ar1 *milliseconds*
sets the autorepeat delay (length of time in milliseconds that a key must be depressed before autorepeat starts).

–ar2 *milliseconds*
sets the autorepeat interval (length of time in milliseconds that should elapse between autorepeat-generated keystrokes).

–noloadxkb
disables loading of an XKB keymap description on server startup.

–xkbdb *filename*
uses *filename* for default keyboard keymaps.

–xkbmap *filename*
loads keyboard description in *filename* on server startup.

SECURITY EXTENSION OPTIONS

X servers that support the SECURITY extension accept the following option:

–sp *filename*
causes the server to attempt to read and interpret filename as a security policy file with the format described below. The file is read at server startup and reread at each server reset.

The syntax of the security policy file is as follows. Notation: "*" means zero or more occurrences of the preceding element, and "+" means one or more occurrences. To interpret <foo/bar>, ignore the text after the /; it is used to distinguish between instances of <foo> in the next section.

<policy file> ::= <version line> <other line>*

<version line> ::= <string/v> '\n'

<other line > ::= <comment> | <access rule> | <site policy> | <blank line>

<comment> ::= # <not newline>* '\n'

<blank line> ::= <space> '\n'

<site policy> ::= sitepolicy <string/sp> '\n'

<access rule> ::= property <property/ar> <window> <perms> '\n'

<property> ::= <string>

<window> ::= any | root | <required property>

<required property> ::= <property/rp> | <property with value>

<property with value> ::= <property/rpv> = <string/rv>

<perms> ::= [<operation> | <action> | <space>]*

<operation> ::= r | w | d

<action> ::= a | i | e

<string> ::= <dbl quoted string> | <single quoted string> | <unqouted string>

<dbl quoted string> ::= <space> " <not dqoute>* " <space>

<single quoted string> ::= <space> ' <not squote>* ' <space>

<unquoted string> ::= <space> <not space>+ <space>

<space> ::= [' ' | '\t']*

Character sets:

<not newline> ::= any character except '\n'
<not dqoute> ::= any character except "
<not squote> ::= any character except '
<not space> ::= any character except those in <space>

The semantics associated with the above syntax are as follows.

<version line>, the first line in the file, specifies the file format version. If the server does not recognize the version <string/v>, it ignores the rest of the file. The version string for the file format described here is "version-1" .

Once past the <version line>, lines that do not match the above syntax are ignored.

<comment> lines are ignored.

<sitepolicy> lines are currently ignored. They are intended to specify the site policies used by the XC-QUERY-SECURITY-1 authorization method.

<access rule> lines specify how the server should react to untrusted client requests that affect the X Window property named <property/ar>. The rest of this section describes the interpretation of an <access rule>.

For an <access rule> to apply to a given instance of <property/ar>, <property/ar> must be on a window that is in the set of windows specified by <window>. If <window> is any, the rule applies to <property/ar> on any window. If <window> is root, the rule applies to <property/ar> only on root windows.

If <window> is <required property>, the following apply. If <required property> is a <property/rp>, the rule applies when the window also has that <property/rp>, regardless of its value. If <required property> is a <property with value>, <property/rpv> must also have the value specified by <string/rv>. In this case, the property must have type STRING and format 8, and should contain one or more null-terminated strings. If any of the strings match <string/rv>, the rule applies.

The definition of string matching is simple case-sensitive string comparison with one elaboration: the occurrence of the character '*' in <string/rv> is a wildcard meaning "any string." A <string/rv> can contain multiple wildcards anywhere in the string. For example, "x*" matches strings that begin with x, "*x" matches strings that end with x, "*x*" matches strings containing x, and "x*y*" matches strings that start with x and subsequently contain y.

There may be multiple <access rule> lines for a given <property/ar>. The rules are tested in the order that they appear in the file. The first rule that applies is used.

<perms> specify operations that untrusted clients may attempt, and the actions that the server should take in response to those operations.

<operation> can be r (read), w (write), or d (delete). The following table shows how X Protocol property requests map to these operations in The Open Group server implementation.

GetProperty	r, or r and d if delete = True
ChangeProperty	w
RotateProperties	r and w
DeleteProperty	d
ListProperties	none, untrusted clients can always list all properties

<action> can be a (allow), i (ignore), or e (error). Allow means execute the request as if it had been issued by a trusted client. Ignore means treat the request as a no-op. In the case of GetProperty, ignore means return an empty property value if the property exists, regardless of its actual value. Error means do not execute the request and return a BadAtom error with the atom set to the property name. Error is the default action for all properties, including those not listed in the security policy file.

An <action> applies to all <operation>s that follow it, until the next <action> is encountered. Thus, irwad means ignore read and write, allow delete.

GetProperty and RotateProperties may do multiple operations (r and d, or r and w). If different actions apply to the operations, the most severe action is applied to the whole request; there is no partial request execution. The severity ordering is: allow < ignore < error. Thus, if the <perms> for a property are ired (ignore read, error delete), and an untrusted client attempts GetProperty on that property with delete = True, an error is returned, but the property value is not. Similarly, if any of the properties in a RotateProperties do not allow both read and write, an error is returned without changing any property values.

Here is an example security policy file.

```
version-1

# Allow reading of application resources, but not writing.
property RESOURCE_MANAGER           root        ar iw
property SCREEN_RESOURCES           root        ar iw

# Ignore attempts to use cut buffers.  Giving errors causes apps to crash,
# and allowing access may give away too much information.
property CUT_BUFFER0                root        irw
property CUT_BUFFER1                root        irw
property CUT_BUFFER2                root        irw
property CUT_BUFFER3                root        irw
property CUT_BUFFER4                root        irw
property CUT_BUFFER5                root        irw
property CUT_BUFFER6                root        irw
property CUT_BUFFER7                root        irw

# If you are using Motif, you probably want these.
property _MOTIF_DEFAULT_BINDINGS    root        ar iw
property _MOTIF_DRAG_WINDOW         root        ar iw
property _MOTIF_DRAG_TARGETS        any         ar iw
property _MOTIF_DRAG_ATOMS          any         ar iw
property _MOTIF_DRAG_ATOM_PAIRS     any         ar iw

# The next two rules let xwininfo -tree work when untrusted.
property WM_NAME                    any         ar

# Allow read of WM_CLASS, but only for windows with WM_NAME.
# This might be more restrictive than necessary, but demonstrates
# the <required property> facility, and is also an attempt to
# say "top level windows only."
property WM_CLASS                   WM_NAME     ar
```

```
# These next three let xlsclients work untrusted.  Think carefully
# before including these; giving away the client machine name and command
# may be exposing too much.
property WM_STATE                          WM_NAME         ar
property WM_CLIENT_MACHINE                 WM_NAME         ar
property WM_COMMAND                        WM_NAME         ar

# To let untrusted clients use the standard colormaps created by
# xstdcmap, include these lines.
property RGB_DEFAULT_MAP                   root            ar
property RGB_BEST_MAP                      root            ar
property RGB_RED_MAP                       root            ar
property RGB_GREEN_MAP                     root            ar
property RGB_BLUE_MAP                      root            ar
property RGB_GRAY_MAP                      root            ar

# To let untrusted clients use the color management database created
# by xcmsdb, include these lines.
property XDCCC_LINEAR_RGB_CORRECTION       root            ar
property XDCCC_LINEAR_RGB_MATRICES         root            ar
property XDCCC_GRAY_SCREENWHITEPOINT       root            ar
property XDCCC_GRAY_CORRECTION             root            ar

# To let untrusted clients use the overlay visuals that many vendors
# support, include this line.
property SERVER_OVERLAY_VISUALS            root            ar

# Dumb examples to show other capabilities.

# oddball property names and explicit specification of error conditions
property "property with spaces"            'property with "'  aw er ed

# Allow deletion of Woo-Hoo if window also has property OhBoy with value
# ending in "son".  Reads and writes will cause an error.
property Woo-Hoo                           OhBoy = "*son"  ad
```

NETWORK CONNECTIONS

The X server supports client connections via a platform-dependent subset of the following transport types: TCPIP, Unix Domain sockets, DECnet, and several varieties of SVR4 local connections. See the DISPLAY NAMES section of the *X*(7) manual page to learn how to specify which transport type clients should try to use.

GRANTING ACCESS

The X server implements a platform-dependent subset of the following authorization protocols: MIT-MAGIC-COOKIE-1, XDM-AUTHORIZATION-1, XDM-AUTHORIZATION-2, SUN-DES-1, and MIT-KERBEROS-5. See the *Xsecurity*(7) manual page for information on the operation of these protocols.

Authorization data required by the above protocols 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. See *xauth*(1) for maintenance of this file, and distribution of its contents to remote hosts.

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::) or a complete name in the format *family*:*name* as described in the *xhost*(1) manual page. There should be no leading or trailing spaces on any lines. For example:

```
joesworkstation
corporate.company.com
star::
inet:bigcpu
local:
```

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.

If the X FireWall Proxy (*xfwp*) is being used without a sitepolicy, host-based authorization must be turned on for clients to be able to connect to the X server via the *xfwp*. If *xfwp* is run without a configuration file and thus no sitepolicy is defined, if *xfwp* is using an X server where xhost + has been run to turn off host-based authorization checks, when a client tries to connect to this X server via *xfwp*, the X server will deny the connection. See *xfwp*(1) for more information about this proxy.

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. X servers that support the SECURITY extension fare better because clients can be designated untrusted via the authorization they use to connect; see the *xauth*(1) manual page for details. Restrictions are imposed on untrusted clients that curtail the mischief they can do. See the SECURITY extension specification for a complete list of these restrictions.

Sites that have better authentication and authorization systems 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

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*.

The default font path is __default_font_path__ .

The font path can be set with the **−fp** option or by *xset*(1) after the server has started.

FILES

*/etc/X***n***.hosts* Initial access control list for display number **n**

/usr/X11R6/lib/X11/fonts/misc,*/usr/X11R6/lib/X11/fonts/75dpi*,*/usr/X11R6/lib/X11/fonts/100dpi*
Bitmap font directories

/usr/X11R6/lib/X11/fonts/TTF,*/usr/X11R6/lib/X11/fonts/Speedo*,*/usr/X11R6/lib/X11/fonts/Type1*
Outline font directories

/usr/X11R6/lib/X11/rgb.txt Color database

*/tmp/.X11-unix/X***n** Unix domain socket for display number **n**

*/tmp/rcX***n** Kerberos 5 replay cache for display number **n**

*/usr/adm/X***n***msgs* Error log file for display number **n** if run from *init*(8)

/usr/X11R6/lib/X11/xdm/xdm-errors
Default error log file if the server is run from *xdm*(1)

SEE ALSO

General information: *X*(7)

Protocols: *X Window System Protocol, The X Font Service Protocol, X Display Manager Control Protocol*

Fonts: *bdftopcf*(1), *mkfontdir*(1), *mkfontscale*(1), *xfs*(1), *xlsfonts*(1), *xfontsel*(1), *xfd*(1), *X Logical Font Description Conventions*

Security: *Xsecurity*(7), *xauth*(1), *Xau*(1), *xdm*(1), *xhost*(1), *xfwp*(1), *Security Extension Specification*

Starting the server: *xdm*(1), *xinit*(1)

Controlling the server once started: *xset*(1), *xsetroot*(1), *xhost*(1)

Server-specific man pages: *Xdec*(1), *XmacII*(1), *Xsun*(1), *Xnest*(1), *Xvfb*(1), *XFree86*(1), *XDarwin*(1).

Server internal documentation: *Definition of the Porting Layer for the X v11 Sample Server*

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. Dave Wiggins took over post-R5 and made substantial improvements.

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*]] [[+-]dpms] [dpms *standby* [*suspend* [*off*]]] [dpms force standby/suspend/off/on] [[-+]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 [keycode]] [r on/off] [r rate *delay* [*rate*]] [s [*length* [*period*]]] [s blank/noblank] [s expose/noexpose] [s on/off] [s default] [s activate] [s reset] [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(7)*.

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.

−dpms
: The **−dpms** option disables DPMS (Energy Star) features.

+dpms
: The **+dpms** option enables DPMS (Energy Star) features.

dpms *flags...*
: The **dpms** option allows the DPMS (Energy Star) parameters to be set. The option can take up to three numerical values, or the 'force' flag followed by a DPMS state. The 'force' flags forces the server to immediately switch to the DPMS state specified. The DPMS state can be one of 'standby', 'suspend', 'off', or 'on'. When numerical values are given, they set the inactivity period (in units of seconds) before the three modes are activated. The first value given is for the 'standby' mode, the second is for the 'suspend' mode, and the third is for the 'off' mode. Setting these values implicitly enables the DPMS features. A value of zero disables a particular mode.

fp= *path,...*
: The **fp=** sets the font path to the entries given in the path argument. The entries are interpreted by the server, not by the client. Typically they are directory names or font server names, but the interpretation is server-dependent.

fp default
The **default** argument causes the font path to be reset to the server's default.

fp rehash
The **rehash** argument resets the font path to its current value, causing 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 entries.

+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 entries.

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. If a specific keycode is specified as a parameter, autorepeat for that keycode is enabled or disabled. If the server supports the XFree86-Misc extension, or the XKB extension, then a parameter of 'rate' is accepted and should be followed by zero, one or two numeric values. The first specifies the delay before autorepeat starts and the second specifies the repeat rate. In the case that the server supports the XKB extension, the delay is the number of milliseconds before autorepeat starts, and the rate is the number of repeats per second. If the rate or delay is not given, it will be set to the default value.

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, an 'activate/reset' 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 'activate' flag forces activation of screen saver even if the screen saver had been turned off. The 'reset' flag forces deactivation of screen saver if it is active. 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(7), Xserver(1), xmodmap(1), xrdb(1), xsetroot(1)

AUTHOR

Bob Scheifler, MIT Laboratory for Computer Science
David Krikorian, MIT Project Athena (X11 version)
XFree86-Misc support added by David Dawes and Joe Moss

NAME

xsetmode − set the mode for an X Input device

SYNOPSIS

xsetmode *device-name* **ABSOLUTE | RELATIVE**

DESCRIPTION

Xsetmode sets the mode of an XInput device to either absolute or relative. This isn't appropriate for all device types.

AUTHOR

Frederic Lepied

NAME

xsetpointer − set an X Input device as the main pointer

SYNOPSIS

xsetpointer −l | *device-name*

DESCRIPTION

Xsetmode sets an XInput device up as the main pointer. When called with the −l flag it lists the available devices.

AUTHOR

Frederic Lepied

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(7)*.

SEE ALSO

X(7), xset(1), xrdb(1)

AUTHOR

Mark Lillibridge, MIT Project Athena

NAME

xsm − X Session Manager

SYNOPSIS

xsm [-display display] [-session sessionName] [-verbose]

DESCRIPTION

xsm is a session manager. A session is a group of applications, each of which has a particular state. *xsm* allows you to create arbitrary sessions - for example, you might have a "light" session, a "development" session, or an "xterminal" session. Each session can have its own set of applications. Within a session, you can perform a "checkpoint" to save application state, or a "shutdown" to save state and exit the session. When you log back in to the system, you can load a specific session, and you can delete sessions you no longer want to keep.

Some session managers simply allow you to manually specify a list of applications to be started in a session. *xsm* is more powerful because it lets you run applications and have them automatically become part of the session. On a simple level, *xsm* is useful because it gives you this ability to easily define which applications are in a session. The true power of *xsm*, however, can be taken advantage of when more and more applications learn to save and restore their state.

OPTIONS

−display *display*
Causes *xsm* to connect to the specified X display.

−session *sessionName*
Causes *xsm* to load the specified session, bypassing the session menu.

−verbose
Turns on debugging information.

SETUP

.xsession file

Using *xsm* requires a change to your *.xsession* file:

The last program executed by your *.xsession* file should be *xsm*. With this configuration, when the user chooses to shut down the session using *xsm*, the session will truly be over.

Since the goal of the session manager is to restart clients when logging into a session, your .xsession file, in general, should not directly start up applications. Rather, the applications should be started within a session. When *xsm* shuts down the session, *xsm* will know to restart these applications. Note however that there are some types of applications that are not "session aware". *xsm* allows you to manually add these applications to your session (see the section titled *Client List*).

SM_SAVE_DIR environment variable

If the *SM_SAVE_DIR* environment variable is defined, *xsm* will save all configuration files in this directory. Otherwise, they will be stored in the user's home directory. Session aware applications are also encouraged to save their checkpoint files in the *SM_SAVE_DIR* directory, although the user should not depend on this convention.

Default Startup Applications

The first time *xsm* is started, it will need to locate a list of applications to start up. For example, this list might include a window manager, a session management proxy, and an xterm. *xsm* will first look for the file *.xsmstartup* in the user's home directory. If that file does not exists, it will look for the *system.xsm* file that was set up at installation time. Note that *xsm* provides a "fail safe" option when the user chooses a session to start up. The fail safe option simply loads the default applications described above.

Each line in the startup file should contain a command to start an application. A sample startup file might look this:

<start of file>
twm

smproxy
xterm
<end of file>

STARTING A SESSION

When *xsm* starts up, it first checks to see if the user previously saved any sessions. If no saved sessions exist, *xsm* starts up a set of default applications (as described above in the section titled *Default Startup Applications*). If at least one session exists, a session menu is presented. The **[-session sessionName]** option forces the specified session to be loaded, bypassing the session menu.

The session menu

The session menu presents the user with a list of sessions to choose from. The user can change the currently selected session with the mouse, or by using the up and down arrows on the keyboard. Note that sessions which are locked (i.e. running on a different display) can not be loaded or deleted.

The following operations can be performed from the session menu:

Load Session Pressing this button will load the currently selected session. Alternatively, hitting the Return key will also load the currently selected session, or the user can double click a session from the list.

Delete Session This operation will delete the currently selected session, along with all of the application checkpoint files associated with the session. After pressing this button, the user will be asked to press the button a second time in order to confirm the operation.

Default/Fail Safe *xsm* will start up a set of default applications (as described above in the section titled *Default Startup Applications*). This is useful when the user wants to start a fresh session, or if the session configuration files were corrupted and the user wants a "fail safe" session.

Cancel Pressing this button will cause *xsm* to exit. It can also be used to cancel a "Delete Session" operation.

CONTROLLING A SESSION

After *xsm* determines which session to load, it brings up its main window, then starts up all applications that are part of the session. The title bar for the session manager's main window will contain the name of the session that was loaded.

The following options are available from *xsm*'s main window:

Client List Pressing this button brings up a window containing a list of all clients that are in the current session. For each client, the host machine that the client is running on is presented. As clients are added and removed from the session, this list is updated to reflect the changes. The user is able to control how these clients are restarted (see below).

By pressing the **View Properties** button, the user can view the session management properties associated with the currently selected client.

By pressing the **Clone** button, the user can start a copy of the selected application.

By pressing the **Kill Client** button, the user can remove a client from the session.

By selecting a restart hint from the **Restart Hint** menu, the user can control the restarting of a client. The following hints are available:

– The **Restart If Running** hint indicates that the client should be restarted in the next session if it is connected to the session manager at the end of the current session.

– The **Restart Anyway** hint indicates that the client should be restarted in the next session even if it exits before the current session is terminated.

– The **Restart Immediately** hint is similar to the **Restart Anyway** hint, but in addition, the client is meant to run continuously. If the client exits, the session manager will try to restart it in the current session.

– The **Restart Never** hint indicates that the client should not be restarted in the next session.

Note that all X applications may not be "session aware". Applications that are not session aware are ones that do not support the X Session Management Protocol or they can not be detected by the Session Management Proxy (see the section titled *THE PROXY*). *xsm* allows the user to manually add such applications to the session. The bottom of the *Client List* window contains a text entry field in which application commands can be typed in. Each command should go on its own line. This information will be saved with the session at checkpoint or shutdown time. When the session is restarted, *xsm* will restart these applications in addition to the regular "session aware" applications.

Pressing the **Done** button removes the **Client List** window.

Session Log... The Session Log window presents useful information about the session. For example, when a session is restarted, all of the restart commands will be displayed in the log window.

Checkpoint By performing a checkpoint, all applications that are in the session are asked to save their state. Not every application will save its complete state, but at a minimum, the session manager is guaranteed that it will receive the command required to restart the application (along with all command line options). A window manager participating in the session should guarantee that the applications will come back up with the same window configurations.

If the session being checkpointed was never assigned a name, the user will be required to specify a session name. Otherwise, the user can perform the checkpoint using the current session name, or a new session name can be specified. If the session name specified already exists, the user will be given the opportunity to specify a different name or to overwrite the already existing session. Note that a session which is locked can not be overwritten.

When performing a checkpoint, the user must specify a **Save Type** which informs the applications in the session how much state they should save.

The **Local** type indicates that the application should save enough information to restore the state as seen by the user. It should not affect the state as seen by other users. For example, an editor would create a temporary file containing the contents of its editing buffer, the location of the cursor, etc...

The **Global** type indicates that the application should commit all of its data to permanent, globally accessible storage. For example, the editor would simply save the edited file.

The **Both** type indicates that the application should do both of these. For example, the editor would save the edited file, then create a temporary file with information such as the location of the cursor, etc...

In addition to the **Save Type**, the user must specify an **Interact Style**.

The **None** type indicates that the application should not interact with the user while saving state.

The **Errors** type indicates that the application may interact with the user only if an error condition arises.

The **Any** type indicates that the application may interact with the user for any purpose. Note that *xsm* will only allow one application to interact with the user at a time.

After the checkpoint is completed, *xsm* will, if necessary, display a window containing the list of applications which did not report a successful save of state.

Shutdown A shutdown provides all of the options found in a checkpoint, but in addition, can cause the session to exit. Note that if the interaction style is **Errors** or **Any**, the user may cancel the shutdown. The user may also cancel the shutdown if any of the applications report an unsuccessful save of state.

The user may choose to shutdown the session with our without performing a checkpoint.

HOW *XSM* RESPONDS TO SIGNALS

xsm will respond to a SIGTERM signal by performing a shutdown with the following options: fast, no interaction, save type local. This allows the user's session to be saved when the system is being shutdown. It can also be used to perform a remote shutdown of a session.

xsm will respond to a SIGUSR1 signal by performing a checkpoint with the following options: no interaction, save type local. This signal can be used to perform a remote checkpoint of a session.

THE PROXY

Since not all applications have been ported to support the X Session Management Protocol, a proxy service exists to allow "old" clients to work with the session manager. In order for the proxy to detect an application joining a session, one of the following must be true:

- The application maps a top level window containing the **WM_CLIENT_LEADER** property. This property provides a pointer to the client leader window which contains the **WM_CLASS**, **WM_NAME**, **WM_COMMAND**, and **WM_CLIENT_MACHINE** properties.

or ...

- The application maps a top level window which does not contain the **WM_CLIENT_LEADER** property. However, this top level window contains the **WM_CLASS**, **WM_NAME**, **WM_COMMAND**, and **WM_CLIENT_MACHINE** properties.

An application that support the **WM_SAVE_YOURSELF** protocol will receive a **WM_SAVE_YOURSELF** client message each time the session manager issues a checkpoint or shutdown. This allows the application to save state. If an application does not support the **WM_SAVE_YOURSELF** protocol, then the proxy will provide enough information to the session manager to restart the application (using **WM_COMMAND**), but no state will be restored.

REMOTE APPLICATIONS

xsm requires a remote execution protocol in order to restart applications on remote machines. Currently, *xsm* supports the *rstart* protocol. In order to restart an application on remote machine **X**, machine **X** must have *rstart* installed. In the future, additional remote execution protocols may be supported.

SEE ALSO

smproxy(1), rstart(1)

AUTHORS

Ralph Mor, X Consortium
Jordan Brown, Quarterdeck Office Systems

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 specific standard colormap property, or all such properties, should be removed. *map* may be one of: default, best, red, green, blue, gray, or all.

−display *display*
This option specifies the host and display to use; see *X(7)*.

−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(7)

AUTHOR

Donna Converse, MIT X Consortium

NAME

Xsun, XsunMono, Xsun24 − 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

XsunMono supports the BW2 monochrome frame buffer. *Xsun* supports the CG2, CG3, CG4, and CG6 8-bit color frame buffers in addition to the BW2 monochrome frame buffer. On Solaris 2.5 it also supports the TCX as an 8-bit color frame buffer. *Xsun24* supports the cgeight 24-bit color frame buffer in addition to the 8-bit color and monochrome frame buffers that *Xsun* supports.

If specific framebuffer device files aren't specified on the command line with the *−dev* switch or in the *XDEVICE* environment variable, the server will search for all installed frame buffers and will use all those that it finds.

Finally, if no specific framebuffers are found, the generic framebuffer interface */dev/fb* is used.

KEYBOARDS

Xsun, Xsun24, and XsunMono support the Type-2, Type-3, and many variations of the Type-4 and Type-5 keyboards.

Type-4 and Type-5 keyboards feature a key labeled *AltGraph* which is a mode-shift key. The mode-shift key is used to generate the symbols painted on the fronts of the keys. The mode-shift key works exactly like the *Shift*, *Control*, *Alt*, and *<Meta>* keys.

The ten function keys on the left side of the Type-5 keyboard may be considered as having L1..L10 painted on their fronts. Shift-AltGraph will cause different keysyms to be generated for some keys, e.g. the Type-5 *SysRq* key.

For compatibility with Sun's X11/NeWS server, the F11 and F12 keys may be made to generate the equivalent X11/NeWS keysyms by using mode-switch.

For backwards compatibility, the normal and mode-shifted keysyms for the ten function keys on the left side of Type-4 and Type-5 keyboards may be swapped via command line option. See *-swapLkeys*.

The X LEDs 1..4 correspond to the NumLock, ScrollLock, Compose, and CapsLock LEDs respectively. Pressing the key once turns the corresponding LED on. Pressing the key again turns the LED off. Turning an LED on or off with e.g. 'xset [-]led [1234]' is equivalent to pressing the corresponding key.

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 interval in milliseconds between autorepeats of pressed keys.

−swapLkeys
: Swaps the normal keysyms for the function keys on the left side of Type-4 and Type-5 keyboards with the alternate keysyms, i.e. the keysyms painted on the front of the keys.

−flipPixels
: The normal pixel values for white and black are 0 and 1 respectively. When -flipPixels is specified these values are reversed.

–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[:filename]...*
This option specifies the colon separated names of the framebuffer device files to be used.

–fbinfo This option indicates that the server should enumerate the available frame buffers that it will use.

ENVIRONMENT

XDEVICE
If present, and if no explicit **-dev** options are given, specifies the (colon separated) list of display devices to use.

SEE ALSO

X(7), 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**.

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, Kaleb Keithley

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/VT220 (VTxxx) and Tektronix 4014 compatible terminals for programs that cannot 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 VTxxx 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 VTxxx window, and the "Tek Options" menu in the 4014 window.

EMULATIONS

The VT102 emulation is fairly complete, but does not support autorepeat. Double-size characters are displayed properly if your font server supports scalable fonts. Blinking characters are partially implemented; the emulation is functional but does not have the appearance of a real VT102. The VT220 emulation does not support soft fonts, it is otherwise complete. *Termcap*(5) entries that work with *xterm* include an optional platform-specific entry, "xterm," "vt102," "vt100" and "ansi," and "dumb." *xterm* automatically searches the termcap file in this order for these entries and then sets the "TERM" and the "TERMCAP" environment variables. You may also use "vt220," but must set the terminal emulation level with the **decTerminalID** resource. (The "TERMCAP" environment variable is not set if *xterm* is linked against a terminfo library, since the requisite information is not provided by the termcap emulation of terminfo libraries).

Many of the special *xterm* features 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. It supports 12-bit graphics addressing, scaled to the window size. Four different font sizes and five different lines types are supported. There is no write-through or defocused mode support. 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***yyyy***−***MM***−***dd***.***hh***:***mm***:***ss*", where *yyyy*, *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*).

Not all of the features described in this manual are necessarily available in this version of xterm. Some (e.g., the non-VT220 extensions) are available only if they were compiled in, though the most commonly-used are in the default configuration.

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. A popup menu entry makes it simple to switch between the normal and alternate screens for cut and paste.

In either VT102 or Tektronix mode, there are escape sequences to change the name of the windows. Additionally, in VT102 mode, *xterm* implements the window-manipulation control sequences from *dtterm*, such as resizing the window, setting its location on the screen.

Xterm allows character-based applications to receive mouse events (currently button-press and release events, and button-motion events) as keyboard control sequences. See *Xterm Control Sequences* for details.

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). The **−version** and **−help** options are interpreted even if *xterm* cannot open the display, and are useful for testing and configuration scripts:

−version
: This causes *xterm* to print a version number to the standard output.

−help
: This causes *xterm* to print out a verbose message describing its options. The message is written to the standard error.

The other options are used to control the appearance and behavior. Not all options are necessarily configured into your copy of *xterm*.

−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.

−ai
: This option disables active icon support if that feature was compiled into xterm. This is equivalent to setting the *vt100* resource **activeIcon** to FALSE.

+ai
: This option enables active icon support if that feature was compiled into xterm. This is equivalent to setting the *vt100* resource **activeIcon** to TRUE.

−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.

−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.

+bc
: turn off text cursor blinking. This overrides the **cursorBlink** resource.

−bc
: turn on text cursor blinking. This overrides the **cursorBlink** resource.

−bcf *milliseconds*
: time text cursor is off when blinking

−bcn *milliseconds*
: time text cursor is on when blinking

−bdc
: Set the *vt100* resource **colorBDMode** to FALSE, disabling the display of characters with bold attribute as color

+bdc
: Set the *vt100* resource **colorBDMode** to TRUE, enabling the display of characters with bold attribute as color rather than bold

−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.

−cjk_width
Set the **cjkWidth** resource to ‘‘true’’. When turned on, characters with East Asian Ambiguous (A) category in UTR 11 have a column width of 2. Othrwise, they have a column width of 1. This may be useful for some legacy CJK text terminal-based programs assuming box drawings and others to have a column width of 2. It also has to be turned on when you specify a truetype CJK double-width (bi-width/monospace) font either with **−fa** at the command line or **faceName** resource. The default is ‘‘false’’

+cjk_width
Reset the **cjkWidth** resource.

−class *string*
This option allows you to override *xterm*’s resource class. Normally it is ‘‘XTerm’’, but can be set to another class such as ‘‘UXTerm’’ to override selected resources.

−cm This option disables recognition of ANSI color-change escape sequences.

+cm This option enables recognition of ANSI color-change escape sequences. This is the same as the *vt100* resource **colorMode**.

−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 *more*(1) program that causes it to incorrectly display lines that are exactly the width of the window and are followed by a line beginning with a tab (the leading tabs are not displayed). This option is so named because it was originally thought to be a bug in the *curses*(3x) cursor motion package.

+cu This option indicates that *xterm* should not work around the *more*(1) bug mentioned above.

−dc This option disables the escape sequence to change dynamic colors: the vt100 foreground and background colors, the text cursor color, the mouse cursor foreground and background colors, the Tektronix emulator foreground and background colors, and highlight color.

+dc This option enables the escape sequence to change dynamic colors.

−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.**

−en *encoding*
This option determines the encoding on which *xterm* runs. It corresponds to the **locale** resource. Encodings other than UTF-8 are supported by using *luit*. The **−lc** option should be used instead of **−en** for systems with locale support.

−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. See also the discussion of **boldFont** and **boldMode** resources.

−fa *pattern*
This option sets the pattern for fonts selected from the FreeType library if support for that library was compiled into xterm. This corresponds to the **faceName** resource. When a CJK double-width font is specified, you also need to turn on the **cjkWidth** resource.

−fbb This option indicates that *xterm* should compare normal and bold fonts bounding boxes to ensure they are compatible.

+fbb This option indicates that *xterm* should not compare normal and bold fonts bounding boxes to ensure they are compatible.

−fbx This option indicates that *xterm* should not assume that the normal and bold fonts have VT100 line-drawing characters. If any are missing, xterm will draw the characters directly.

+fbx This option indicates that *xterm* should assume that the normal and bold fonts have VT100 line-drawing characters.

−fi *font* This option sets the font for active icons if that feature was compiled into xterm. See also the discussion of the **iconFont** resource.

−fs *size* This option sets the pointsize for fonts selected from the FreeType library if support for that library was compiled into xterm. This corresponds to the **faceSize** resource.

−fw *font* This option specifies the font to be used for displaying wide text. By default, it will attempt to use a font twice as wide as the font that will be used to draw normal text. If no doublewidth font is found, it will improvise, by stretching the normal font. This corresponds to the **wideFont** resource.

−fwb *font*
This option specifies the font to be used for displaying bold wide text. By default, it will attempt to use a font twice as wide as the font that will be used to draw bold text. If no doublewidth font is found, it will improvise, by stretching the bold font. This corresponds to the **wideBoldFont** resource.

−fx *font* This option specifies the font to be used for displaying the preedit string in the "OverTheSpot" input method. See also the discussion of the **ximFont** resource.

−hc *color*
This option specifies the color to use for the background of selected or otherwise highlighted text. If not specified, reverse video is used.

−hf This option indicates that HP Function Key escape codes should be generated for function keys.

+hf This option indicates that HP Function Key escape codes should not be generated for function keys.

−hold Turn on the **hold** resource, i.e., xterm will not immediately destroy its window when the shell command completes. It will wait until you use the window manager to destroy/kill the window, or if you use the menu entries that send a signal, e.g., HUP or KILL.

+hold Turn off the **hold** resource, i.e., xterm will immediately destroy its window when the shell command completes.

−ie Turn on the **ptyInitialErase** resource, i.e., use the pseudo-terminal's sense of the stty erase value.

+ie Turn off the **ptyInitialErase** resource, i.e., set the stty erase value using the **kb** string from the termcap entry as a reference, if available.

−im Turn on the **useInsertMode** resource.

+im Turn off the **useInsertMode** resource.

−into *windowId*
Given an X window identifier (a decimal integer), xterm will reparent its top-level shell widget to that window. This is used to embed xterm within other applications.

−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 does not fall as far behind. Its use is strongly recommended since it makes *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.

−k8 This option sets the **allowC1Printable** resource. When **allowC1Printable** is set, *xterm* overrides the mapping of C1 control characters (code 128-159) to treat them as printable.

+k8 This option resets the **allowC1Printable** resource.

−l Turn logging on. Normally logging is not supported, due to security concerns. Some versions of *xterm* may have logging enabled. The logfile is written to the directory from which xterm is invoked. The filename is generated, of the form

XtermLog.*XXXXXX*

or

Xterm.log.*hostname.yyyy.mm.dd.hh.mm.ss.XXXXXX*

depending on how xterm was built.

+l Turn logging off.

−lc Turn on support of various encodings according to the users’ locale setting, i.e., LC_ALL, LC_CTYPE, or LANG environment variables. This is achieved by turning on UTF-8 mode and by invoking *luit* for conversion between locale encodings and UTF-8. (*luit* is not invoked in UTF-8 locales.) This corresponds to the **locale** resource.

The actual list of encodings which are supported is determined by *luit*. Consult the *luit* manual page for further details. See also the discussion of the **−u8** option which supports UTF-8 locales.

+lc Turn off support of automatic selection of locale encodings. Conventional 8bit mode or, in UTF-8 locales or with **−u8** option, UTF-8 mode will be used.

−lcc *path*
File name for the encoding converter from/to locale encodings and UTF-8 which is used with **−lc** option or **locale** resource. This corresponds to the **localeFilter** resource.

−leftbar Force scrollbar to the left side of VT100 screen. This is the default, unless you have set the rightScrollBar resource.

−lf *filename*
Specify the log-filename. See the **−l** option.

−ls This option indicates that the shell that is started in the *xterm* window will 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).

The **−ls** flag and the **loginShell** resource are ignored if **−e** is also given, because *xterm* does not know how to make the shell start the given command after whatever it does when it is a login shell − the user’s shell of choice need not be a Bourne shell after all. Also, *xterm −e* is supposed to provide a consistent functionality for other applications that need to start text-mode programs in a window, and if **loginShell** were not ignored, the result of ˜/.profile might interfere with that.

If you do want the effect of **−ls** and **−a** simultaneously, you may get away with something like

```
xterm -e /bin/bash -l -c "my command here"
```

Finally, **−ls** is not completely ignored, because *xterm −ls −e* does write a */etc/wtmp* entry (if configured to do so), whereas *xterm −e* does not.

+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.

−mesg Turn off the **messages** resource, i.e., disallow write access to the terminal.

+mesg Turn on the **messages** resource, i.e., allow write access to the terminal.

−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.

−nul This option disables the display of underlining.

+nul This option enables the display of underlining.

−pc This option enables the PC-style use of bold colors (see boldColors resource).

+pc This option disables the PC-style use of bold colors.

−pob This option indicates that the window should be raised whenever a Control-G is received.

+pob This option indicates that the window should not be raised whenever a Control-G is received.

−rightbar
Force scrollbar to the right side of VT100 screen.

−rvc This option disables the display of characters with reverse attribute as color.

+rvc This option enables the display of characters with reverse attribute as color.

−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.

−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.

−samename
Does not send title and icon name change requests when the request would have no effect: the name is not changed. This has the advantage of preventing flicker and the disadvantage of requiring an extra round trip to the server to find out the previous value. In practice this should never be a problem.

+samename
Always send title and icon name change requests.

−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.

−sm This option, corresponding to the *sessionMgt* resource, indicates that *xterm* should set up session manager callbacks.

+sm This option indicates that *xterm* should not set up session manager callbacks.

−sp This option indicates that Sun/PC keyboard should be assumed, providing mapping for keypad '+' to ',', and CTRL-F1 to F13, CTRL-F2 to F14, etc.

+sp This option indicates that the standard escape codes should be generated for keypad and function keys.

−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. *Termcap*(5) entries that work with *xterm* ''tek4014,'' ''tek4015,'' ''tek4012'', ''tek4013'' and ''tek4010,'' and ''dumb.'' *xterm* automatically searches the termcap file in this order for these entries and then sets the ''TERM'' and the ''TERMCAP'' environment variables.

+t This option indicates that *xterm* should start in VT102 mode.

−ti *term_id*
Specify the name used by *xterm* to select the correct response to terminal ID queries. It also specifies the emulation level, used to determine the type of response to a DA control sequence. Valid values include vt52, vt100, vt101, vt102, and vt220 (the "vt" is optional). The default is vt100. The term_id argument specifies the terminal ID to use. (This is the same as the *decTerminalID* resource).

−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 ^char (e.g., ^c or ^u) and ^? may be used to indicate delete (127).

−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.

−u8 This option sets the **utf8** resource. When **utf8** is set, xterm interprets incoming data as UTF-8. This sets the **wideChars** resource as a side-effect, but the UTF-8 mode set by this option prevents it from being turned off. If you must turn it on and off, use the **wideChars** resource.

This option and the **utf8** resource are overridden by the **−lc** and **−en** options and **locale** resource. That is, if *xterm* has been compiled to support *luit*, and the **locale** resource is not ''false'' this option is ignored. We recommend using the **−lc** option or the ‘‘**locale: true**’’ resource in UTF-8 locales when your operating system supports locale, or **−en UTF-8** option or the ‘‘**locale: UTF-8**’’ resource when your operating system does not support locale.

+u8 This option resets the **utf8** resource.

−ulc This option disables the display of characters with underline attribute as color rather than with underlining.

+ulc This option enables the display of characters with underline attribute as color rather than with underlining.

−ut This option indicates that *xterm* should not write a record into the the system *utmp* log file.

+ut This option indicates that *xterm* should write a record into the system *utmp* log file.

−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.

−wc This option sets the **wideChars** resource. When **wideChars** is set, xterm maintains internal structures for 16-bit characters. If you do not set this resource to ‘‘true’’, xterm will ignore the escape sequence which turns UTF-8 mode on and off. The default is ‘‘false’’.

+wc This option resets the **wideChars** resource.

−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.

−ziconbeep *percent*
Same as zIconBeep resource. If percent is non-zero, xterms that produce output while iconified will cause an XBell sound at the given volume and have "***" prepended to their icon titles. Most window managers will detect this change immediately, showing you which window has the output. (A similar feature was in x10 xterm.)

−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 allows *xterm* to be used as an input and output channel for an existing program and is sometimes used in specialized applications. The option value specifies the last few letters of the name of a pseudo-terminal to use in slave mode, plus the number of the inherited file descriptor. If the option contains a ‘‘/’’ character, that delimits the characters used for the pseudo-terminal name from the file descriptor. Otherwise, exactly two characters are used from the option for the pseudo-terminal name, the remainder is the file descriptor. Examples:

```
-S123/45
-Sab34
```

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*:

−bd *color*
: This option specifies the color to use for the border of the window. The default is ‘‘black.’’

−bg *color*
: This option specifies the color to use for the background of the window. The default is ‘‘white.’’

−bw *number*
: This option specifies the width in pixels of the border surrounding the window.

−display *display*
: This option specifies the X server to contact; see *X(7)*.

−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*.

−geometry *geometry*
: This option specifies the preferred size and position of the VT102 window; see *X(7)*.

−iconic This option indicates that *xterm* should ask the window manager to start it as an icon rather than as the normal window.

−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.

−rv This option indicates that reverse video should be simulated by swapping the foreground and background colors.

+rv Disable the simulation of reverse video by swapping foreground and background colors.

−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.

−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.

RESOURCES

The program understands all of the core X Toolkit resource names and classes. Application specific resources (e.g., "**XTerm.***NAME*") follow:

backarrowKeyIsErase (class **BackarrowKeyIsErase)**
: Tie the VTxxx **backarrowKey** and **ptyInitialErase** resources together by setting the DECBKM state according to whether the initial value of stty erase is a backspace (8) or delete (127) character. The default is ‘‘false’’, which disables this feature.

hold (class **Hold)**
If true, xterm will not immediately destroy its window when the shell command completes. It will wait until you use the window manager to destroy/kill the window, or if you use the menu entries that send a signal, e.g., HUP or KILL. You may scroll back, select text, etc., to perform most graphical operations. Resizing the display will lose data, however, since this involves interaction with the shell which is no longer running.

hpFunctionKeys (class **HpFunctionKeys)**
Specifies whether or not HP Function Key escape codes should be generated for function keys instead of standard escape sequences.

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.

messages (class **Messages)**
Specifies whether write access to the terminal is allowed initially. See **mesg**(1). The default is ‘‘true’’.

ptyHandshake (class**PtyHandshake)**
If ‘‘true’’, *xterm* will perform handshaking during initialization to ensure that the parent and child processes update the **utmp** and **stty** state. Platforms with newer pseudo-terminal interfaces do not require this feature; normally it is not configured. The default is ‘‘true’’.

ptyInitialErase (class **PtyInitialErase)**
If ‘‘true’’, *xterm* will use the pseudo-terminal’s sense of the stty erase value. If ‘‘false’’, *xterm* will set the stty erase value to match its own configuration, using the **kb** string from the termcap entry as a reference, if available. In either case, the result is applied to the TERMCAP variable which *xterm* sets. The default is ‘‘false’’.

sameName (class **SameName)**
If the value of this resource is ‘‘true’’, xterm does not send title and icon name change requests when the request would have no effect: the name is not changed. This has the advantage of preventing flicker and the disadvantage of requiring an extra round trip to the server to find out the previous value. In practice this should never be a problem. The default is ‘‘true’’.

sessionMgt (class **SessionMgt)**
If the value of this resource is ‘‘true’’, xterm sets up session manager callbacks for **XtNdieCallback** and **XtNsaveCallback**. The default is ‘‘true’’.

sunFunctionKeys (class **SunFunctionKeys)**
Specifies whether or not Sun Function Key escape codes should be generated for function keys instead of standard escape sequences.

sunKeyboard (class **SunKeyboard)**
Specifies whether or not Sun/PC keyboard layout should be assumed rather than DEC VT220. This causes the keypad ‘+’ to be mapped to ‘,’. and CTRL F1-F12 to F11-F20, depending on the setting of the **ctrlFKeys** resource. so *xterm* emulates a DEC VT220 more accurately. Otherwise (the default, with **sunKeyboard** set to ‘‘false’’), *xterm* uses PC-style bindings for the function keys and keypad.

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, lnext and status. Control characters may be specified as ^char (e.g., ^c or ^u) and **^?** may be used to indicate delete (127). Use **^-** to denote undef. Use **\034** to represent **^**, since a literal backslash in an X resource escapes the next character.

This is very useful for overriding the default terminal settings without having to do an *stty* every time an *xterm* is started. Note, however, that the stty program on a given host may use different keywords; *xterm*'s table is builtin.

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 the system *utmp* log file.

waitForMap (class **WaitForMap)**
Specifies whether or not *xterm* should wait for the initial window map before starting the subprocess. The default is "false."

zIconBeep (class **ZIconBeep)**
Same as −ziconbeep command line argument. If the value of this resource is non-zero, xterms that produce output while iconified will cause an XBell sound at the given volume and have "***" prepended to their icon titles. Most window managers will detect this change immediately, showing you which window has the output. (A similar feature was in x10 *xterm*.) The default is "false."

The following resources are specified as part of the *vt100* widget (class *VT100*): These are specified by patterns such as "**XTerm.vt100.***NAME*":

activeIcon (class **ActiveIcon)**
Specifies whether or not active icon windows are to be used when the *xterm* window is iconified, if this feature is compiled into *xterm*. The active icon is a miniature representation of the content of the window and will update as the content changes. Not all window managers necessarily support application icon windows. Some window managers will allow you to enter keystrokes into the active icon window. The default is "false."

allowC1Printable (class **AllowC1Printable)**
If true, overrides the mapping of C1 controls (codes 128-159) to make them be treated as if they were printable characters. Although this corresponds to no particular standard, some users insist it is a VT100. The default is "false."

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. The default is "false."

allowWindowOps (class **AllowWindowOps)**
Specifies whether extended window control sequences (as used in dtterm) for should be allowed. The default is "true."

answerbackString (class **AnswerbackString)**
Specifies the string that *xterm* sends in response to an ENQ (control/E) character from the host. The default is a blank string, i.e., "". A hardware VT100 implements this feature as a setup option.

alwaysHighlight (class **AlwaysHighlight)**
Specifies whether or not *xterm* should always display a highlighted text cursor. By default (if this resource is false), a hollow text cursor is displayed whenever the pointer moves out of the

window or the window loses the input focus. The default is ‘‘false.’’

alwaysUseMods (class **AlwaysUseMods**)
Override the **numLock** resource, telling *xterm* to use the Alt and Meta modifiers as to construct parameters for function key sequences even if those modifiers appear in the translations resource. The default is ‘‘false.’’

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.’’

awaitInput (class **AwaitInput**)
Specifies whether or not the *xterm* uses a 50 millisecond timeout to await input (i.e., to support the Xaw3d arrow scrollbar). The default is ‘‘false.’’

backarrowKey (class **BackarrowKey**)
Specifies whether the backarrow key transmits a backspace (8) or delete (127) character. This corresponds to the DECBKM control sequence. The default (backspace) is ‘‘true.’’ Pressing the control key toggles this behavior.

background (class **Background**)
Specifies the color to use for the background of the window. The default is ‘‘white.’’

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.

boldColors (class **ColorMode**)
Specifies whether to combine bold attribute with colors like the IBM PC, i.e., map colors 0 through 7 to colors 8 through 15. These normally are the brighter versions of the first 8 colors, hence bold. The default is ‘‘true.’’

boldFont (class **BoldFont**)
Specifies the name of the bold font to use instead of overstriking. There is no default for this resource.

boldMode (class **BoldMode**)
This specifies whether or not text with the bold attribute should be overstruck to simulate bold fonts if the resolved bold font is the same as the normal font. It may be desirable to disable bold fonts when color is being used for the bold attribute. Note that *xterm* has one bold font which you may set explicitly. It attempts to match a bold font for the other font selections (**font1** through **font6**). If the normal and bold fonts are distinct, this resource has no effect. The default is ‘‘true.’’

Although *xterm* attempts to match a bold font for other font selections, the font server may not cooperate. Since X11R6, bitmap fonts have been scaled. The font server claims to provide the bold font that *xterm* requests, but the result is not always readable. XFree86 provides a feature which can be used to suppressed the scaling. In the X server’s configuration file (e.g., "/etc/X11/XFree86"), you can add ":unscaled" to the end of the directory specification for the "misc" fonts, which comprise the fixed-pitch fonts that are used by xterm. For example

```
FontPath"/usr/lib/X11/fonts/misc/"
```

would become

```
FontPath"/usr/lib/X11/fonts/misc/:unscaled"
```

Depending on your configuration, the font server may have its own configuration file. The same ":unscaled" can be added to its configuration file at the end of the directory specification for "misc".

brokenLinuxOSC (class **BrokenLinuxOSC)**

If true, *xterm* applies a workaround to ignore malformed control sequences that a Linux script might send. Compare the palette control sequences documented in *console_codes* with ECMA-48.

brokenSelections (class **BrokenSelections)**

If true, *xterm* in 8-bit mode will interpret **STRING** selections as carrying text in the current locale's encoding. Normally **STRING** selections carry ISO-8859-1 encoded text. Setting this resource to ‘‘true’’ violates the ICCCM; it may, however, be useful for interacting with some broken X clients. The default is ‘‘false.’’

brokenStringTerm (class **BrokenStringTerm)**

provides a work-around for some ISDN routers which start an application control string without completing it. Set this to ‘‘true’’ if *xterm* appears to freeze when connecting. The default is ‘‘false.’’

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.’’

cacheDoublesize (class **CacheDoublesize)**

Specifies the maximum number of double-sized fonts which are cached by *xterm*. The default (8) may be too large for some X terminals with limited memory. Set this to zero to disable doublesize fonts altogether.

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 CHARACTER CLASSES section.

cjkWidth (class **CjkWidth)**

Specifies whether *xterm* should follow the traditional East Asian width convention. When turned on, characters with East Asian Ambiguous (A) category in UTR 11 have a column width of 2. You may have to set this option to ‘‘true’’ if you have some old East Asian terminal based programs that assume that line-drawing characters have a column width of 2. The default is ‘‘false.’’

curses (class **Curses)**

Specifies whether or not the last column bug in *more*(1) should be worked around. See the **−cu** option for details. The default is ‘‘false.’’

colorAttrMode (class **ColorAttrMode)**

Specifies whether ‘‘colorBD’’, ‘‘colorBL’’, ‘‘colorUL’’, and ‘‘colorRV’’ should override ANSI colors. If not, these are displayed only when no ANSI colors have been set for the corresponding position. The default is ‘‘false.’’

colorMode (class **ColorMode)**

Specifies whether or not recognition of ANSI (ISO 6429) color change escape sequences should be enabled. The default is ‘‘true.’’

colorBDMode (class **ColorAttrMode)**
Specifies whether characters with the bold attribute should be displayed in color or as bold characters. Note that setting **colorMode** off disables all colors, including bold. The default is ‘‘false.’’

colorBLMode (class **ColorAttrMode)**
Specifies whether characters with the blink attribute should be displayed in color. Note that setting **colorMode** off disables all colors, including this. The default is ‘‘false.’’

colorRVMode (class **ColorAttrMode)**
Specifies whether characters with the reverse attribute should be displayed in color. Note that setting **colorMode** off disables all colors, including this. The default is ‘‘false.’’

colorULMode (class **ColorAttrMode)**
Specifies whether characters with the underline attribute should be displayed in color or as underlined characters. Note that setting **colorMode** off disables all colors, including underlining. The default is ‘‘false.’’

color0 (class **Color0)**

color1 (class **Color1)**

color2 (class **Color2)**

color3 (class **Color3)**

color4 (class **Color4)**

color5 (class **Color5)**

color6 (class **Color6)**

color7 (class **Color7)**
These specify the colors for the ISO 6429 extension. The defaults are, respectively, black, red3, green3, yellow3, DodgerBlue1, magenta3, cyan3, and gray90. The default shades of color are chosen to allow the colors 8-15 to be used as brighter versions.

color8 (class **Color8)**

color9 (class **Color9)**

color10 (class **Color10)**

color11 (class **Color11)**

color12 (class **Color12)**

color13 (class **Color13)**

color14 (class **Color14)**

color15 (class **Color15)**
These specify the colors for the ISO 6429 extension if the bold attribute is also enabled. The default resource values are, respectively, gray30, red, green, yellow, SteelBlue1, magenta, cyan, and white.

color16 (class **Color16)**

through

color255 (class **Color255)**
These specify the colors for the 256-color extension. The default resource values are for colors 16 through 231 to make a 6x6x6 color cube, and colors 232 through 255 to make a grayscale ramp.

colorBD (class **ColorBD)**
This specifies the color to use to display bold characters if the ‘‘colorBDMode’’ resource is enabled. The default is ‘‘XtDefaultForeground.’’

colorBL (class **ColorBL)**
This specifies the color to use to display blink characters if the ‘‘colorBLMode’’ resource is enabled. The default is ‘‘XtDefaultForeground.’’

colorRV (class **ColorRV)**
This specifies the color to use to display reverse characters if the ‘‘colorRVMode’’ resource is enabled. The default is ‘‘XtDefaultForeground.’’

colorUL (class **ColorUL)**
This specifies the color to use to display underlined characters if the ‘‘colorULMode’’ resource is enabled. The default is ‘‘XtDefaultForeground.’’

ctrlFKeys (class **CtrlFKeys)**
In VT220 keyboard mode (see **sunKeyboard** resource), specifies the amount by which to shift F1-F12 given a control modifier (CTRL). This allows you to generate key symbols for F10-F20 on a Sun/PC keyboard. The default is ‘‘10’’, which means that CTRL F1 generates the key symbol for F11.

cursorBlink (class **CursorBlink)**
Specifies whether to make the cursor blink. The default is ‘‘false.’’

cursorColor (class **CursorColor)**
Specifies the color to use for the text cursor. The default is ‘‘black.’’

cursorOffTime (class **CursorOffTime)**
Specifies the duration of the "off" part of the cursor blink cycle-time in milliseconds. The default is 300.

cursorOnTime (class **CursorOnTime)**
Specifies the duration of the "on" part of the cursor blink cycle-time, in milliseconds. The default is 600.

highlightColor (class **HighlightColor)**
Specifies the color to use for the background of selected or otherwise highlighted text. If not specified, reverse video is used. The default is ‘‘XtDefaultForeground.’’

decTerminalID (class **DecTerminalID)**
Specifies the emulation level (100=VT100, 220=VT220, etc.), used to determine the type of response to a DA control sequence. The default is 100.

deleteIsDEL (class **DeleteIsDEL)**
Specifies whether the Delete key on the editing keypad should send DEL (127) or the VT220-style Remove escape sequence. The default is ‘‘false,’’ for the latter.

dynamicColors (class **DynamicColors)**
Specifies whether or not escape sequences to change colors assigned to different attributes are recognized.

eightBitControl (class **EightBitControl**)
Specifies whether or not control sequences sent by the terminal should be eight-bit characters or escape sequences. The default is ‘‘false.’’

eightBitInput (class **EightBitInput**)
If ‘‘true’’, Meta characters input from the keyboard are presented as a single character with the eighth bit turned on. The terminal is put into 8-bit mode. If ‘‘false’’, Meta characters are converted into a two-character sequence with the character itself preceded by ESC. The terminal is put into 7-bit mode. The **metaSendsEscape** resource may override this. 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,’’ which means that they are accepted as is.

faceName (class **FaceName)**
Specify the pattern for fonts selected from the FreeType library if support for that library was compiled into *xterm*. There is no default. If not specified, or if there is no match for both normal and bold fonts, *xterm* uses the **font** and related resources.

faceSize (class **FaceSize)**
Specify the pointsize for fonts selected from the FreeType library if support for that library was compiled into *xterm*. The default is ‘‘14.’’

font (class **Font)**
Specifies the name of the normal font. The default is ‘‘fixed.’’

See the discussion of the **locale** resource, which describes how this font may be overridden.

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.

fontDoublesize (class **FontDoublesize)**
Specifies whether *xterm* should attempt to use font scaling to draw doublesize characters. Some older font servers cannot do this properly, will return misleading font metrics. The default is ‘‘true’’. If disabled, *xterm* will simulate doublesize characters by drawing normal characters with spaces between them.

forceBoxChars (class **Boolean)**
Specifies whether *xterm* should assume the normal and bold fonts have VT100 line-drawing characters. If ‘‘false’’, *xterm* will check for missing characters in the 1-31 cells and make line-drawing characters directly. The default is ‘‘false.’’

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.’’

freeBoldBox (class **Boolean)**
Specifies whether *xterm* should assume the bounding boxes for normal and bold fonts are compatible. If ‘‘false’’, *xterm* compares them and will reject choices of bold fonts that do not match the size of the normal font. The default is ‘‘false’’, which means that the comparison is performed.

geometry (class **Geometry)**
Specifies the preferred size and position of the VT102 window. There is no default for this resource.

highlightSelection (class **HighlightSelection)**
If ‘‘false’’, selecting with the mouse highlights all positions on the screen between the beginning of the selection and the current position. If ‘‘true’’, *xterm* highlights only the positions that contain text that can be selected. The default is ‘‘false.’’

Depending on the way your applications write to the screen, there may be trailing blanks on a line. *Xterm* stores data as it is shown on the screen. Erasing the display changes the internal state of each cell so it is not considered a blank for the purpose of selection. Blanks written since the last erase are selectable. If you do not wish to have trailing blanks in a selection, use the **trimSelection** resource.

hpLowerleftBugCompat (class **HpLowerleftBugCompat)**
Specifies whether to work around a bug in HP's *xdb*, which ignores termcap and always sends ESC F to move to the lower left corner. ''true'' causes *xterm* to interpret ESC F as a request to move to the lower left corner of the screen. The default is ''false.''

i18nSelections (class **I18nSelections)**
If false, *xterm* will never request the targets **COMPOUND_TEXT** or **TEXT**. The default is ''true.'' It may be set to false in order to work around ICCCM violations by other X clients.

iconBorderColor (class **BorderColor)**
Specifies the border color for the active icon window if this feature is compiled into *xterm*. Not all window managers will make the icon border visible.

iconBorderWidth (class **BorderWidth)**
Specifies the border width for the active icon window if this feature is compiled into *xterm*. The default is 0 (no border). Not all window managers will make the border visible.

iconFont (class **IconFont)**
Specifies the font for the miniature active icon window, if this feature is compiled into *xterm*. The default is "nil2".

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.''

keyboardDialect (class **KeyboardDialect)**
Specifies the initial keyboard dialect, as well as the default value when the terminal is reset. The value given is the same as the final character in the control sequences which change character sets. The default is ''B'', which corresponds to US ASCII.

keymap*NAME* (class Keymap*NAME*)
See the discussion of the **keymap()** action.

limitResize (class **LimitResize)**
Limits resizing of the screen via control sequence to a given multiple of the display dimensions. The default is ''1''.

locale (class **Locale)**
Specifies how to use *luit*, an encoding converter between UTF-8 and locale encodings. The resource value (ignoring case) may be:

true
xterm will use the encoding specified by the users' LC_CTYPE locale (i.e., LC_ALL, LC_CTYPE, or LANG variables) as far as possible. This is realized by always enabling UTF-8 mode and invoking *luit* in non-UTF-8 locales.

medium
xterm will follow users' LC_CTYPE locale only for UTF-8, east Asian, and Thai locales, where the encodings were not supported by conventional 8bit mode with changing fonts. For other locales, *xterm* will use conventional 8bit mode.

no *xterm* will use conventional 8bit mode or UTF-8 mode according to **utf8** resource or **–u8** option.

Any other value, e.g., “UTF-8” or “ISO8859-2”, is assumed to be an encoding name; *luit* will be invoked to support the encoding. The actual list of supported encodings depends on *luit*. The default is “medium”.

Regardless of your locale and encoding, you need an ISO-10646-1 font to display the result. Your configuration may not include this font, or locale-support by *xterm* may not be needed. At startup, *xterm* uses a mechanism equivalent to the **load-vt-fonts(utf8Fonts, Utf8Fonts)** action to load font name subresources of the VT100 widget. That is, resource patterns such as "***vt100.utf8Fonts.font**" will be loaded, and (if this resource is enabled), override the normal fonts. If no subresources are found, the normal fonts such as "***vt100.font**", etc., are used. The resource files distributed with *xterm* use ISO-10646-1 fonts, but do not rely on them unless you are using the locale mechanism.

localeFilter (class **LocaleFilter)**
Specifies the file name for the encoding converter from/to locale encodings and UTF-8 which is used with the **−lc** option or **locale** resource. The help message shown by “xterm −help” lists the default value, which depends on your system configuration.

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.”

metaSendsEscape (class **MetaSendsEscape**)
If “true”, Meta characters are converted into a two-character sequence with the character itself preceded by ESC. This applies as well to function key control sequences, unless *xterm* sees that **Meta** is used in your key translations. If “false”, Meta characters input from the keyboard are handled according to the **eightBitInput** resource. The default is “false.”

modifyCursorKeys (class **ModifyCursorKeys**)
Tells how to handle the special case where control-, shift-, alt- or meta-modifiers are used to add a parameter to the escape sequence returned by a cursor-key. Set it to 0 to use the old/obsolete behavior. Set it to 1 to prefix modified sequences with CSI. Set it to 2 to force the modifier to be the second parameter. Set it to 3 to mark the sequence with a ’>’ to hint that it is private. The default is “2”.

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.

numLock (class **NumLock)**
If “true”, *xterm* checks if NumLock is used as a modifier (see *xmodmap*(1)). If so, this modifier is used to simplify the logic when implementing special NumLock for the **sunKeyboard** resource. Also (when **sunKeyboard** is false), similar logic is used to find the modifier associated with the left and right Alt keys. The default is “true.”

oldXtermFKeys (class **OldXtermFKeys)**
If “true”, *xterm* will use old-style control sequences for function keys F1 to F4, for compatibility with X Consortium *xterm*. Otherwise, it uses the VT100-style codes for PF1 to PF4. The default is “false.”

pointerColor (class **PointerColor)**
Specifies the foreground color of the pointer. The default is ‘‘XtDefaultForeground.’’

pointerColorBackground (class **PointerColorBackground)**
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.’’

popOnBell (class **PopOnBell)**
Specifies whether the window whould be raised when Control-G is received. The default is ‘‘false.’’

printAttributes (class **PrintAttributes)**
Specifies whether to print graphic attributes along with the text. A real DEC VTxxx terminal will print the underline, highlighting codes but your printer may not handle these. A ‘‘0’’ disables the attributes. A ‘‘1’’ prints the normal set of attributes (bold, underline, inverse and blink) as VT100-style control sequences. A ‘‘2’’ prints ANSI color attributes as well. The default is ‘‘1.’’

printerAutoClose (class **PrinterAutoClose)**
If ‘‘true’’, *xterm* will close the printer (a pipe) when the application switches the printer offline with a Media Copy command. The default is ‘‘false.’’

printerCommand (class **PrinterCommand)**
Specifies a shell command to which *xterm* will open a pipe when the first MC (Media Copy) command is initiated. The default is ‘‘lpr.’’ If the resource value is given as a blank string, the printer is disabled.

printerControlMode (class **PrinterControlMode)**
Specifies the printer control mode. A ‘‘1’’ selects autoprint mode, which causes *xterm* to print a line from the screen when you move the cursor off that line with a line feed, form feed or vertical tab character, or an autowrap occurs. Autoprint mode is overridden by printer controller mode (a ‘‘2’’), which causes all of the output to be directed to the printer. The default is ‘‘0.’’

printerExtent (class **PrinterExtent)**
Controls whether a print page function will print the entire page (true), or only the the portion within the scrolling margins (false). The default is ‘‘false.’’

printerFormFeed (class **PrinterFormFeed)**
Controls whether a form feed is sent to the printer at the end of a print page function. The default is ‘‘false.’’

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.’’

rightScrollBar (class **RightScrollBar)**
Specifies whether or not the scrollbar should be displayed on the right rather than the left. 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.’’

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.

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.’’

shiftFonts (class **ShiftFonts**)
Specifies whether to enable the actions **larger-vt-font()** and **smaller-vt-font()**, which are normally bound to the shifted KP_Add and KP_Subtract. The default is ‘‘true.’’

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. There is no default for this resource.

tekInhibit (class **TekInhibit**)
Specifies whether or not the escape sequence to enter Tektronix mode should be ignored. 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 *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. *Xterm* supports terminfo in a different way, supporting composite control sequences (also known as private modes) 1047, 1048 and 1049 which have the same effect as the original 47 control sequence. The default for this resource is ‘‘false.’’

tiXtraScroll (class **TiXtraScroll**)
Specifies whether *xterm* should scroll to a new page when processing the *ti* termcap entry, i.e., the private modes 47, 1047 or 1049. This is only in effect if *titeInhibit* is ‘‘true’’, because the intent of this option is to provide a picture of the full-screen application’s display on the scrollback without wiping out the text that would be shown before the application was initialized. The default for this resource is ‘‘false.’’

translations (class **Translations**)
Specifies the key and button bindings for menus, selections, ‘‘programmed strings,’’ etc. See the **ACTIONS** section.

trimSelection (class **TrimSelection)**
If you set **highlightSelection**, you can see the text which is selected, including any trailing spaces. Clearing the screen (or a line) resets it to a state containing no spaces. Some lines may contain trailing spaces when an application writes them to the screen. However, you may not wish to paste lines with trailing spaces. If this resource is true, *xterm* will trim trailing spaces from text which is selected. It does not affect spaces which result in a wrapped line, nor will it trim the trailing newline from your selection. The default is ‘‘false.’’

underLine (class **UnderLine)**
This specifies whether or not text with the underline attribute should be underlined. It may be desirable to disable underlining when color is being used for the underline attribute. The default is ‘‘true.’’

utf8 (class **Utf8)**
This specifies whether *xterm* will run in UTF-8 mode. If you set this resource, *xterm* also sets the **wideChars** resource as a side-effect. When set via a resource, *xterm* cannot be switched via control sequences out of UTF-8 mode. The default is ‘‘0’’ (off). Any other value will turn on UTF-8 mode. See the **locale** resource for non-UTF-8 locales.

utf8Fonts (class **Utf8Fonts)**
See the discussion of the **locale** resource.

veryBoldColors (class **VeryBoldColors)**
Specifies whether to combine video attributes with colors specified by **colorBD**, **colorBL** and **colorUL**. The resource value is the sum of values for each attribute: 2 for underline, 4 for bold and 8 for blink. The default is ‘‘0.’’

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.’’

visualBellDelay (class **VisualBellDelay)**
Number of milliseconds to delay when displaying a visual bell. Default is 100. If set to zero, no visual bell is displayed. This is useful for very slow displays, e.g., an LCD display on a laptop.

vt100Graphics (class **VT100Graphics)**
This specifies whether *xterm* will interpret VT100 graphic character escape sequences while in UTF-8 mode. The default is ‘‘true’’, to provide support for various legacy applications.

wideBoldFont (class **WideBoldFont)**
This option specifies the font to be used for displaying bold wide text. By default, it will attempt to use a font twice as wide as the font that will be used to draw bold text. If no doublewidth font is found, it will improvise, by stretching the bold font.

wideChars (class **WideChars)**
Specifies if *xterm* should respond to control sequences that process 16-bit characters. The default is ‘‘false.’’

wideFont (class **WideFont)**
This option specifies the font to be used for displaying wide text. By default, it will attempt to use a font twice as wide as the font that will be used to draw normal text. If no doublewidth font is found, it will improvise, by stretching the normal font.

ximFont (class **XimFont)**
This option specifies the font to be used for displaying the preedit string in the "OverTheSpot" input method.

In "OverTheSpot" preedit type, the preedit (preconversion) string is displayed at the position of the cursor. It is the XIM server’s responsibility to display the preedit string. The XIM client must inform the XIM server of the cursor position. For best results, the preedit string must be displayed with a proper font. Therefore, *xterm* informs the XIM server of the proper font. The font is be supplied by a "fontset", whose default value is "*". This matches every font, the X library automatically chooses fonts with proper charsets. The **ximFont** resource is provided to override this default font setting.

The following resources are specified as part of the *tek4014* widget (class *Tek4014*). These are specified by patterns such as "**XTerm.tek4014.***NAME*":

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.

fontLarge (class **Font)**
Specifies the large font to use in the Tektronix window.

fontSmall (class **Font)**
Specifies the small font to use in the Tektronix window.

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.’’

height (class **Height)**
Specifies the height of the Tektronix window in pixels.

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.’’

width (class **Width)**
Specifies the width of the Tektronix window in pixels.

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. Resources named "**line***N*" where *N* is a number are separators with class **SmeLine**.

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.

redraw (class **SmeBSB)**
This entry invokes the **redraw()** action.

logging (class **SmeBSB)**
This entry invokes the **logging(toggle)** action.

print (class **SmeBSB)**
This entry invokes the **print()** action.

print-redir (class **SmeBSB)**
This entry invokes the **print-redir()** action.

8-bit-control (class **SmeBSB)**
This entry invokes the **set-8-bit-control(toggle)** action.

backarrow key (class **SmeBSB)**
This entry invokes the **set-backarrow(toggle)** action.

num-lock (class **SmeBSB)**
This entry invokes the **set-num-lock(toggle)** action.

meta-esc (class **SmeBSB)**
This entry invokes the **meta-sends-escape(toggle)** action.

delete-is-del (class **SmeBSB)**
This entry invokes the **delete-is-del(toggle)** action.

oldFunctionKeys (class **SmeBSB)**
This entry invokes the **old-function-keys(toggle)** action.

hpFunctionKeys (class **SmeBSB)**
This entry invokes the **hp-function-keys(toggle)** action.

scoFunctionKeys (class **SmeBSB)**
This entry invokes the **sco-function-keys(toggle)** action.

sunFunctionKeys (class **SmeBSB)**
This entry invokes the **sun-function-keys(toggle)** action.

sunKeyboard (class **SmeBSB)**
This entry invokes the **sunKeyboard(toggle)** action.

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.

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.

poponbell (class **SmeBSB)**
This entry invokes the **set-poponbell(toggle)** action.

marginbell (class **SmeBSB)**
This entry invokes the **set-marginbell(toggle)** action.

cursorblink (class **SmeBSB)**
This entry invokes the **set-cursorblink(toggle)** action.

titeInhibit (class **SmeBSB)**
This entry invokes the **set-titeInhibit(toggle)** action.

activeicon (class **SmeBSB)**
This entry toggles active icons on and off if this feature was compiled into *xterm*. It is enabled only if *xterm* was started with the command line option +ai or the **activeIcon** resource is set to ‘‘True.’’

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.

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.

altscreen (class **SmeBSB)**
This entry invokes the **set-altscreen(toggle)** 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.

font-linedrawing (class **SmeBSB)**
This entry invokes the **set-font-linedrawing(s)** action.

font-doublesize (class **SmeBSB)**
This entry invokes the **set-font-doublesize(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.

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.

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. Logical words and lines selected by double- or triple-clicking may wrap across more than one screen line if lines were wrapped by *xterm* itself rather than by the application running in the window. 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 does 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. Most 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 a series of comma-separated of *range*:*value* pairs. The *range* is either a single number or *low*-*high* in the range of 0 to 65535, corresponding to the 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. When not in UTF-8 mode, only the first 256 bytes of this table will be used.

The default table starts as follows -

```
static int charClass[256] = {
/* 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,
/* x80  x81  x82  x83  IND  NEL  SSA  ESA */
     1,   1,   1,   1,   1,   1,   1,   1,
/* HTS  HTJ  VTS  PLD  PLU   RI  SS2  SS3 */
     1,   1,   1,   1,   1,   1,   1,   1,
/* DCS  PU1  PU2  STS  CCH   MW  SPA  EPA */
     1,   1,   1,   1,   1,   1,   1,   1,
/* x98  x99  x9A  CSI   ST  OSC   PM  APC */
     1,   1,   1,   1,   1,   1,   1,   1,
/*   -    i   c/    L   ox   Y-    |   So */
   160, 161, 162, 163, 164, 165, 166, 167,
/*  ..   c0   ip   <<    _         R0    - */
   168, 169, 170, 171, 172, 173, 174, 175,
/*   o   +-    2    3    '    u   q|    . */
   176, 177, 178, 179, 180, 181, 182, 183,
/*   ,    1    2   >>  1/4  1/2  3/4    ? */
   184, 185, 186, 187, 188, 189, 190, 191,
/*  A`   A'   A^   A~   A:   Ao   AE   C, */
```

```
        48,  48,  48,  48,  48,  48,  48,  48,
/*  E`   E'   E^   E:   I`   I'   I^   I: */
        48,  48,  48,  48,  48,  48,  48,  48,
/*  D-   N~   O`   O'   O^   O~   O:    X */
        48,  48,  48,  48,  48,  48,  48, 215,
/*  O/   U`   U'   U^   U:   Y'    P    B */
        48,  48,  48,  48,  48,  48,  48,  48,
/*  a`   a'   a^   a~   a:   ao   ae   c, */
        48,  48,  48,  48,  48,  48,  48,  48,
/*  e`   e'   e^   e:    i`  i'   i^   i: */
        48,  48,  48,  48,  48,  48,  48,  48,
/*   d   n~   o`   o'   o^   o~   o:   -: */
        48,  48,  48,  48,  48,  48,  48, 247,
/*  o/   u`   u'   u^   u:   y'    P   y: */
        48,  48,  48,  48,  48,  48,  48,  48};
```

For example, the string "33:48,37:48,45-47:48,38:48" 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 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:

allow-send-events(*on/off/toggle***)**
: This action set or toggles the **allowSendEvents** resource and is also invoked by the **allowsends** entry in *mainMenu*.

bell([*percent***])**
: This action rings the keyboard bell at the specified percentage above or below the base volume.

clear-saved-lines()
: This action does **hard-reset()** (see below) and also clears the history of lines saved off the top of the screen. It is also invoked from the **clearsavedlines** entry in *vtMenu*. The effect is identical to a hardware reset (RIS) control sequence.

create-menu(*m/v/f/t***)**
: This action creates one of the menus used by *xterm*, if it has not been previously created. The parameter values are the menu names: *mainMenu*, *vtMenu*, *fontMenu*, *tekMenu*, respectively.

deiconify()
: Changes the window state back to normal, if it was iconified.

delete-is-del()
: This action toggles the state of the **deleteIsDEL** resource.

dired-button()
: Handles a button event (other than press and release) by echoing the event's position (i.e., character line and column) in the following format:

^X ESC G <line+' '> <col+' '>

iconify()
: Iconifies the window.

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*.

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-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 **metaSendsEscape** and the **eightBitInput** resources.

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*.

insert-seven-bit()
This action is a synonym for **insert()**

interpret(*control-sequence***)**
Interpret the given control sequence locally, i.e., without passing it to the host. This works by inserting the control sequence at the front of the input buffer. Use "\" to escape octal digits in the string. Xt does not allow you to put a null character (i.e., "\000") in the string.

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.

larger-vt-font()
Set the font to the next larger one, based on the font dimensions. See also **set-vt-font()**.

load-vt-fonts(*name***[,***class***])**
Load fontnames from the given subresource name and class. That is, load the "*VT100.*name*.font", resource as "*VT100.font" etc. If no name is given, the original set of fontnames is restored.

Unlike **set-vt-font()**, this does not affect the escape- and select-fonts, since those are not based on resource values. It does affect the fonts loosely organized under the ‘‘Default’’ menu entry: **font**, **boldFont**, **wideFont** and **wideBoldFont**.

maximize()
Resizes the window to fill the screen.

meta-sends-escape()
This action toggles the state of the **metaSendsEscape** resource.

popup-menu(*menuname***)**
This action displays the specified popup menu. Valid names (case is significant) include: *mainMenu*, *vtMenu*, *fontMenu*, and *tekMenu*.

print() This action prints the window and is also invoked by the *print* entry in *mainMenu*.

print-redir()
This action toggles the **printerControlMode** between 0 and 2. The corresponding popup menu entry is useful for switching the printer off if you happen to change your mind after deciding to print random binary files on the terminal.

quit() This action sends a SIGHUP to the subprogram and exits. It is also invoked by the **quit** entry in *mainMenu*.

redraw()
This action redraws the window and is also invoked by the *redraw* entry in *mainMenu*.

restore()
Restores the window to the size before it was last maximized.

scroll-back(*count* **[,***units* **[,***mouse*] **])**
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.

An adjustment can be specified for these values by appending a "+" or "-" sign followed by a number, e.g., *page-2* to specify 2 lines less than a page.

If the third parameter *mouse* is given, the action is ignored when mouse reporting is enabled.

scroll-forw(*count* **[,***units* **[,***mouse*] **])**
This action scrolls is similar to **scroll-back** except that it scrolls the other direction.

secure() This action toggles the *Secure Keyboard* mode described in the section named **SECURITY**, and is invoked from the **securekbd** entry in *mainMenu*.

select-cursor-end(*destname* **[, ...])**
This action is similar to **select-end** except that it should be used with **select-cursor-start**.

select-cursor-start()
This action is similar to **select-start** except that it begins the selection at the current text cursor position.

select-end(*destname* **[, ...])**
This action puts the currently selected text into all of the selections or cutbuffers specified by *destname*.

select-extend()
This action tracks the pointer and extends the selection. It should only be bound to Motion events.

select-set()
This action stores text that corresponds to the current selection, without affecting the selection mode.

select-start()
This action begins text selection at the current pointer location. See the section on **POINTER USAGE** for information on making selections.

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*.

set-allow132(*on/off/toggle***)**
This action toggles the **c132** resource and is also invoked from the **allow132** entry in *vtMenu*.

set-altscreen(*on/off/toggle***)**
This action toggles between the alternate and current screens.

set-appcursor(*on/off/toggle***)**
This action toggles the handling Application Cursor Key mode and is also invoked by the **appcursor** 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-autolinefeed(*on/off/toggle***)**
This action toggles automatic insertion of linefeeds and is also invoked by the **autolinefeed** 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-backarrow(*on/off/toggle***)**
This action toggles the **backarrowKey** resource and is also invoked from the **backarrow key** entry in *vtMenu*.

set-cursorblink(*on/off/toggle***)**
This action toggles the **cursorBlink** resource and is also invoked from the **cursorblink** 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-font-doublesize(*on/off/toggle***)**
This action toggles the **fontDoublesize** resource and is also invoked by the **font-doublesize** entry in *fontMenu*.

set-hp-function-keys(*on/off/toggle***)**
This action toggles the **hpFunctionKeys** resource and is also invoked by the **hpFunctionKeys** entry in *mainMenu*.

set-jumpscroll(*on/off/toggle***)**
This action toggles the **jumpscroll** resource and is also invoked by the **jumpscroll** entry in *vtMenu*.

set-font-linedrawing(*on/off/toggle***)**
This action toggles the *xterm*'s state regarding whether the current font has line-drawing characters and whether it should draw them directly. It is also invoked by the **font-linedrawing** entry in *fontMenu*.

set-logging()
This action toggles the state of the logging option.

set-old-function-keys(*on/off/toggle***)**
This action toggles the state of legacy function keys and is also invoked by the **oldFunctionKeys** entry in *mainMenu*.

set-marginbell(*on/off/toggle***)**
This action toggles the **marginBell** resource and is also invoked from the **marginbell** entry in *vtMenu*.

set-num-lock()
This action toggles the state of the **numLock** resource.

set-pop-on-bell(*on/off/toggle***)**
This action toggles the **popOnBell** resource and is also invoked by the **poponbell** 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-reversewrap(*on/off/toggle***)**
This action toggles the **reverseWrap** resource and is also invoked by the **reversewrap** 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-scrollbar(*on/off/toggle***)**
This action toggles the **scrollbar** resource and is also invoked by the **scrollbar** entry in *vtMenu*.

set-sco-function-keys(*on/off/toggle***)**
This action toggles the **scoFunctionKeys** resource and is also invoked by the **scoFunctionKeys** entry in *mainMenu*.

set-sun-function-keys(*on/off/toggle***)**
This action toggles the **sunFunctionKeys** resource and is also invoked by the **sunFunctionKeys** entry in *mainMenu*.

set-sun-keyboard(*on/off/toggle***)**
This action toggles the **sunKeyboard** resource and is also invoked by the **sunKeyboard** entry in *mainMenu*.

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*.

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-titeInhibit(*on/off/toggle***)**
This action toggles the **titeInhibit** resource, which controls switching between the alternate and current screens.

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-visual-bell(*on/off/toggle***)**
This action toggles the **visualBell** resource and is also invoked by the **visualbell** entry in *vtMenu*.

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.

If *xterm* is configured to support wide characters, an additional two optional parameters are recognized for the *e* argument: wide font and wide bold font.

smaller-vt-font()
Set the font to the next smaller one, based on the font dimensions. See also **set-vt-font()**.

soft-reset()
This action resets the scrolling region and is also invoked from the **softreset** entry in *vtMenu*. The effect is identical to a soft reset (DECSTR) control sequence.

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.

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*.

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*.

vi-button()
Handles a button event (other than press and release) by echoing a control sequence computed from the event's line number in the screen relative to the current line:

ESC ^P

or

ESC ^N

according to whether the event is before, or after the current line, respectively. The ^N (or ^P) is repeated once for each line that the event differs from the current line. The control sequence is omitted altogether if the button event is on the current line.

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\
     Shift~Ctrl <KeyPress> KP_Add:larger-vt-font() \n\
     Shift Ctrl <KeyPress> KP_Add:smaller-vt-font() \n\
     Shift <KeyPress> KP_Subtract:smaller-vt-font() \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\
  !Lock Ctrl @Num_Lock <Btn1Down>:popup-menu(mainMenu) \n\
      ! @Num_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\
     !Lock Ctrl @Num_Lock <Btn2Down>:popup-menu(vtMenu) \n\
         ! @Num_Lock Ctrl <Btn2Down>:popup-menu(vtMenu) \n\
              ~Ctrl ~Meta <Btn2Down>:ignore() \n\
                     Meta <Btn2Down>:clear-saved-lines() \n\
                ~Ctrl ~Meta <Btn2Up>:insert-selection(PRIMARY, CUT_BUFFER0) \n\
                    !Ctrl <Btn3Down>:popup-menu(fontMenu) \n\
               !Lock Ctrl <Btn3Down>:popup-menu(fontMenu) \n\
     !Lock Ctrl @Num_Lock <Btn3Down>:popup-menu(fontMenu) \n\
         ! @Num_Lock Ctrl <Btn3Down>:popup-menu(fontMenu) \n\
              ~Ctrl ~Meta <Btn3Down>:start-extend() \n\
                  ~Meta <Btn3Motion>:select-extend() \n\
                     Ctrl <Btn4Down>:scroll-back(1,halfpage,m) \n\
                Lock Ctrl <Btn4Down>:scroll-back(1,halfpage,m) \n\
      Lock @Num_Lock Ctrl <Btn4Down>:scroll-back(1,halfpage,m) \n\
           @Num_Lock Ctrl <Btn4Down>:scroll-back(1,halfpage,m) \n\
                          <Btn4Down>:scroll-back(5,line,m)     \n\
                     Ctrl <Btn5Down>:scroll-forw(1,halfpage,m) \n\
                Lock Ctrl <Btn5Down>:scroll-forw(1,halfpage,m) \n\
      Lock @Num_Lock Ctrl <Btn5Down>:scroll-forw(1,halfpage,m) \n\
           @Num_Lock Ctrl <Btn5Down>:scroll-forw(1,halfpage,m) \n\
                          <Btn5Down>:scroll-forw(5,line,m)     \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\
!Lock Ctrl @Num_Lock <Btn1Down>:popup-menu(mainMenu) \n\
     !Ctrl @Num_Lock <Btn1Down>:popup-menu(mainMenu) \n\
               !Ctrl <Btn2Down>: popup-menu(tekMenu) \n\
          !Lock Ctrl <Btn2Down>: popup-menu(tekMenu) \n\
!Lock Ctrl @Num_Lock <Btn2Down>:popup-menu(tekMenu) \n\
     !Ctrl @Num_Lock <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)
```

CONTROL SEQUENCES AND KEYBOARD

The *Xterm Control Sequences* document lists the control sequences which an application can send *xterm* to make it perform various operations. Most of these operations are standardized, from either the DEC or Tektronix terminals, or from more widely used standards such as ISO 6429.

ENVIRONMENT

Xterm sets the environment variables ‘‘TERM’’ for the 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.

Depending on your system configuration, *xterm* may also set the following:

COLUMNS
: the width of the *xterm* in characters (cf: "stty columns").

HOME
: when *xterm* is configured to update utmp.

LINES
: the height of the *xterm* in characters (cf: "stty rows").

LOGNAME
: when *xterm* is configured to update utmp.

SHELL
: when *xterm* is configured to update utmp.

TERMCAP
: the contents of the termcap entry corresponding to $TERM, with lines and columns values substituted for the actual size window you have created.

TERMINFO
: may be defined to a nonstandard location in the configure script.

FILES

The actual pathnames given may differ on your system.

/etc/utmp
: the system logfile, which records user logins.

/etc/wtmp
: the system logfile, which records user logins and logouts.

/usr/X11R6/lib/X11/app-defaults/XTerm
: the *xterm* default application resources.

/usr/X11R6/lib/X11/app-defaults/XTerm-color
: the *xterm* color application resources. If your display supports color, use this

*customization: -color

in your .Xdefaults file to automatically turn on color in *xterm* and similar applications.

ERROR MESSAGES

Most of the fatal error messages from *xterm* use the following format:

xterm: Error *XXX*, errno *YYY*: *ZZZ*

The *XXX* codes (which are used by *xterm* as its exit-code) are listed below, with a brief explanation.

1 is used for miscellaneous errors, usually accompanied by a specific message,

11 ERROR_FIONBIO
main: ioctl() failed on FIONBIO

12 ERROR_F_GETFL
main: ioctl() failed on F_GETFL

13 ERROR_F_SETFL
main: ioctl() failed on F_SETFL

14 ERROR_OPDEVTTY
spawn: open() failed on /dev/tty

15 ERROR_TIOCGETP
spawn: ioctl() failed on TIOCGETP

17 ERROR_PTSNAME
spawn: ptsname() failed

18 ERROR_OPPTSNAME
spawn: open() failed on ptsname

19 ERROR_PTEM
spawn: ioctl() failed on I_PUSH/"ptem"

20 ERROR_CONSEM
spawn: ioctl() failed on I_PUSH/"consem"

21 ERROR_LDTERM
spawn: ioctl() failed on I_PUSH/"ldterm"

22 ERROR_TTCOMPAT
spawn: ioctl() failed on I_PUSH/"ttcompat"

23 ERROR_TIOCSETP
spawn: ioctl() failed on TIOCSETP

24 ERROR_TIOCSETC
spawn: ioctl() failed on TIOCSETC

25 ERROR_TIOCSETD
spawn: ioctl() failed on TIOCSETD

26 ERROR_TIOCSLTC
spawn: ioctl() failed on TIOCSLTC

27 ERROR_TIOCLSET
spawn: ioctl() failed on TIOCLSET

28 ERROR_INIGROUPS
spawn: initgroups() failed

29 ERROR_FORK
spawn: fork() failed

30 ERROR_EXEC
spawn: exec() failed

32 ERROR_PTYS
get_pty: not enough ptys

34 ERROR_PTY_EXEC
waiting for initial map

35 ERROR_SETUID
spawn: setuid() failed

36 ERROR_INIT
spawn: can’t initialize window

46 ERROR_TIOCKSET
spawn: ioctl() failed on TIOCKSET

47 ERROR_TIOCKSETC
spawn: ioctl() failed on TIOCKSETC

48 ERROR_SPREALLOC
spawn: realloc of ttydev failed

49 ERROR_LUMALLOC
luit: command-line malloc failed

50 ERROR_SELECT
in_put: select() failed

54 ERROR_VINIT
VTInit: can't initialize window

57 ERROR_KMMALLOC1
HandleKeymapChange: malloc failed

60 ERROR_TSELECT
Tinput: select() failed

64 ERROR_TINIT
TekInit: can't initialize window

71 ERROR_BMALLOC2
SaltTextAway: malloc() failed

80 ERROR_LOGEXEC
StartLog: exec() failed

83 ERROR_XERROR
xerror: XError event

84 ERROR_XIOERROR
xioerror: X I/O error

90 ERROR_SCALLOC
Alloc: calloc() failed on base

91 ERROR_SCALLOC2
Alloc: calloc() failed on rows

92 ERROR_SREALLOC
ScreenResize: realloc() failed on alt base

96 ERROR_RESIZE
ScreenResize: malloc() or realloc() failed

102 ERROR_SAVE_PTR
ScrnPointers: malloc/realloc() failed

110 ERROR_SBRALLOC
ScrollBarOn: realloc() failed on base

111 ERROR_SBRALLOC2
ScrollBarOn: realloc() failed on rows

121 ERROR_MMALLOC
my_memmove: malloc/realloc failed

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.

This program still needs to be rewritten. It should be split into very modular sections, with the various emulators being completely separate widgets that do not 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 the Tek COPY file name.

SEE ALSO

resize(1), luit(1), X(7), pty(4), tty(4)
Xterm Control Sequences (this is the file ctlseqs.ms).

http://invisible-island.net/xterm/xterm.html

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), Jason Bacon, Stephen P. Wall, David Wexelblat, and Thomas Dickey (XFree86 Project).

NAME

xtrapreset, xtrapinfo, xtrapstats, xtrapout, xtrapin, xtrapchar, xtrapproto - XTrap sample clients

SYNTAX

xtrapreset [**−d[isplay]** *display*]

xtrapinfo [**−d[isplay]** *display*]

xtrapstats [**−d[isplay]** *display*]

xtrapout [**−f** *script*] [**−e**] [**−d[isplay]** *display*] [**−v**]

xtrapin [**−f** *script*] [**−d[isplay]** *display*]

xtrapchar [**−v**] [**−d[isplay]** *display*]

xtrapproto [**−d[isplay]** *display*]

DESCRIPTION

These commands are **SAMPLE CLIENTS** provided with the XTrap X Server Extension Sources, Version 3.3. XTrap is an X Server extension which facilitates the capturing of server protocol and synthesizing core input events. Information on how to obtain these sources can be found in the SOURCES section below.

The **xtrapreset** command is the simplest XTrap client in that it merely performs an XQueryExtension() against XTrap. The name "reset" is historical. The *display* argument is parsed by the X Toolkit and specifies the display where XTrap is to be loaded; see X(1).

xtrapinfo displays general configuration information as a result of an GetAvailable XTrap request to the specified server. It is simply designed to test the request/response mechanism of the XTrap extension and client library as well as display the configuration information that it finds.

xtrapstats tests the event and request vectoring of the server extension by configuring XTrap to collect usage statistics on all core input events and requests. It has a primitive command-line interface for showing the counters, zeroing out the counters, and quitting the program.

xtrapout tests the output transport from the XTrap extension to the XTrap client library. As an aside, since xtrapout has the capability of "recording" events and requests it receives, **xtrapout** is ideal for providing input to test **xtrapin.** Since events are the only concern for the input transport, the −e flag can be specified to indicate that all input events (and only events) should be recorded by **xtrapout.** *script* is specified primarily for non-U*IX machines which don’t support I/O re-direction easily. The −v flag is used to force recording of all requests and input events.

xtrapin is used to test the input transport to the XTrap server extension. As stated earlier, it’s input can be provided by **xtrapout** using the −e qualifier. While it’s primary function is for testing XTrap and serving as an example for XTrap functionality, it can reasonably used as a primitive "playback" client for X sessions.

xtrapchar parses ANSI character sequences including application program sequences to synthesize input events to X Window servers using the XTrap server extension. The intent of this program is to serve as a sample implementation for interfacing character-based alternative input sources into X servers (e.g. voice recognition systems). Another application might be "remote keyboards". The -v flag causes the program to display XTrap configuration information and echo’s characters processed to stdout. If present, this must be the first argument.

Note: **xtrapchar** has only been used with Digital Workstations using the LK201 compatible keyboard. Though reasonable effort was done to maintain portability, no claims are made as to the current level of portability to non-DEC servers for this program.

The **xtrapproto** command is a regression test designed to test the basic XTrap protocol between a client and server. If a given implementation is suspect, the results of this test should be sent to an XTrap implementor and/or developer.

OPTIONS

-d[isplay] *display*

Specifies the server to record from or playback to; see **X(1).**

-e Record only (and all) events. Should be used when creating input for **xtrapin.**

-f *script*
The pathname of the script to be recorded / played back.

-v Verbose mode.

DIAGNOSTICS

X Toolkit Error: Can't load DEC-XTRAP extension

The XTrap X server extension has not been linked into the specified X server.

SOURCES

Sources have been posted on UseNet systems via anonymous ftp.
They are:
East Coast (USA): export@lcs.mit.edu:contrib/XTrap_v32*.tar.Z
West Coast (USA): gatekeeper@pa.dec.com:X11/contrib/XTrap_v32*.tar.Z

IMPORTANT NOTE

Digital participated in the X Consortium's xtest working group which chose to evolve XTrap functionality into a new extension for X11/R6 known as the RECORD extension (XTrap input synthesis functionality is currently covered by the XTEST extension). It is strongly suggested that users of XTrap technology begin developing against RECORD/XTEST as it is the intention of the X Consortium to drive these two extensions in the standards process for providing the protocol capturing/synthesis functionality. Some members of the xtest working group are actively researching migration issues between XTrap and RECORD. If you'd like to contribute, please participate! Contact your local X Consortium Rep for details on how to be added to the xtest mailing list.

If you encounter problems, have questions, etc. with XTrap, please contact via mail, phone, etc. at:

Ken Miller
miller@zk3.dec.com
(VOICE) 603-881-6221
(FAX) 603 881-2257

or paper mail at:

Digital Equipment Corp.
Ken Miller @ ZKO3-3/Y25
110 Spitbrook Rd.
Nashua, NH 03062

Naturally email is preferred and will get the fastest response.

SEE ALSO

X(1)

NAME

xvidtune − video mode tuner for XFree86

SYNOPSIS

xvidtune [**-show** | **-prev** | **-next** | **-unlock**] [−*toolkitoption* . . .]

DESCRIPTION

Xvidtune is a client interface to the XFree86 X server video mode extension (XFree86-VidModeExtension).

When given one of the non-toolkit options, xvidtune provides a command line interface to either switch the video mode.

Without any options (or with only toolkit options) it presents the user with various buttons and sliders that can be used to interactively adjust existing video modes. It will also print the settings in a format suitable for inclusion in an XF86Config file.

Normally the XFree86 X servers only allow changes to be made with the XFree86-VidModeExtension from clients connected via a local connection type.

Note: The original mode settings can be restored by pressing the 'R' key, and this can be used to restore a stable screen in situations where the screen becomes unreadable.

The available buttons are:

Left
Right
Up
Down
Adjust the video mode so that the display will be moved in the appropriate direction.

Wider
Narrower
Shorter
Taller
Adjust the video mode so that the display size is altered appropriately.

Quit Exit the program.

Apply Adjust the current video mode to match the selected settings.

Auto Cause the Up/Down/Right/Left, Wider/Narrower/Shorter/Taller, Restore, and the special S3 buttons to be applied immediately. This button can be toggled.

Test Temporarily switch to the selected settings.

Restore Return the settings to their original values.

Fetch Query the server for its current settings.

Show Print the currently selected settings to stdout in XF86Config "Modeline" format. The primary selection is similarly set.

Next Switch the Xserver to the next video mode.

Prev Switch the Xserver to the previous video mode.

For some S3-based cards (964 and 968) the following are also available:

InvertVCLK
Change the VCLK invert/non-invert state.

EarlySC Change the Early SC state. This affects screen wrapping.

BlankDelay1
BlankDelay2
Set the blank delay values. This affects screen wrapping. Acceptable values are in the range 0−7. The values may be incremented or decremented with the '+' and '-' buttons, or by

pressing the '+' or '-' keys in the text field.

For S3-864/868 based cards *InvertVCLK* and *BlankDelay1* may be useful. For S3 Trio32/Trio64 cards only *InvertVCLK* is available. At the moment there are no default settings available for these chips in the video mode extension and thus this feature is disabled in xvidtune. It can be enabled by setting any of the optional S3 commands in the screen section of XF86Config, e.g. using

blank_delay *"*"* **0**

OPTIONS

xvidtune accepts the standard X Toolkit command line options as well as the following:

−show Print the current settings to stdout in XF86Config "Modeline" format and exit.

−prev Switch the Xserver to the previous video mode.

−next Switch the Xserver to the next video mode.

−unlock Normally, *xvidtune* will disable the switching of video modes via hot-keys while it is running. If for some reason the program did not exit cleanly and they are still disabled, the program can be re-run with this option to re-enable the mode switching key combinations.

SEE ALSO

XF86Config(4/5), XFree86(1)

AUTHORS

Kaleb S. Keithley, X Consortium.
Additions and modifications by Jon Tombs, David Dawes, and Joe Moss.

BUGS

X Error handling, i.e. when the server does not allow xvidtune clients to write new modes, could be better.

NAME

xvinfo - Print out X-Video extension adaptor information

SYNOPSIS

xvinfo [-display *displayname*]

DESCRIPTION

xvinfo prints out the capabilities of any video adaptors associated with the display that are accesible through the X-Video extension.

OPTIONS

-display *display*

This argument allows you to specify the server to query; see *X(7)*.

ENVIRONMENT

DISPLAY

This variable may be used to specify the server to query.

SEE ALSO

xdpyinfo(1)

AUTHORS

Mark Vojkovich

NAME

Xvfb − virtual framebuffer X server for X Version 11

SYNOPSIS

Xvfb [option] ...

DESCRIPTION

Xvfb is an X server that can run on machines with no display hardware and no physical input devices. It emulates a dumb framebuffer using virtual memory.

The primary use of this server was intended to be server testing. The mfb or cfb code for any depth can be exercised with this server without the need for real hardware that supports the desired depths. The X community has found many other novel uses for *Xvfb*, including testing clients against unusual depths and screen configurations, doing batch processing with *Xvfb* as a background rendering engine, load testing, as an aid to porting the X server to a new platform, and providing an unobtrusive way to run applications that don't really need an X server but insist on having one anyway.

BUILDING

To build *Xvfb*, put the following in your host.def and remake.

```
#define BuildServer YES /* if you aren't already building other servers */
#define XVirtualFramebufferServer YES
```

OPTIONS

In addition to the normal server options described in the *Xserver(1)* manual page, *Xvfb* accepts the following command line switches:

−screen *screennum WxHxD*
This option creates screen *screennum* and sets its width, height, and depth to W, H, and D respectively. By default, only screen 0 exists and has the dimensions 1280x1024x8.

−pixdepths *list-of-depths*
This option specifies a list of pixmap depths that the server should support in addition to the depths implied by the supported screens. *list-of-depths* is a space-separated list of integers that can have values from 1 to 32.

−fbdir *framebuffer-directory*
This option specifies the directory in which the memory mapped files containing the framebuffer memory should be created. See FILES. This option only exists on machines that have the mmap and msync system calls.

−shmem
This option specifies that the framebuffer should be put in shared memory. The shared memory ID for each screen will be printed by the server. The shared memory is in xwd format. This option only exists on machines that support the System V shared memory interface.

If neither **−shmem** nor **−fbdir** is specified, the framebuffer memory will be allocated with malloc().

−linebias *n*
This option specifies how to adjust the pixelization of thin lines. The value *n* is a bitmask of octants in which to prefer an axial step when the Bresenham error term is exactly zero. See the file Xserver/mi/miline.h for more information. This option is probably only useful to server developers to experiment with the range of line pixelization possible with the cfb and mfb code.

−blackpixel *pixel-value*, **−whitepixel** *pixel-value*
These options specify the black and white pixel values the server should use.

FILES

The following files are created if the −fbdir option is given.

framebuffer-directory/Xvfb_screen<n>

Memory mapped file containing screen n's framebuffer memory, one file per screen. The file is in xwd format. Thus, taking a full-screen snapshot can be done with a file copy command, and the resulting snapshot will even contain the cursor image.

EXAMPLES

Xvfb :1 -screen 0 1600x1200x32

The server will listen for connections as server number 1, and screen 0 will be depth 32 1600x1200.

Xvfb :1 -screen 1 1600x1200x16

The server will listen for connections as server number 1, will have the default screen configuration (one screen, 1280x1024x8), and screen 1 will be depth 16 1600x1200.

Xvfb -pixdepths 3 27 -fbdir /usr/tmp

The server will listen for connections as server number 0, will have the default screen configuration (one screen, 1280x1024x8), will also support pixmap depths of 3 and 27, and will use memory mapped files in /usr/tmp for the framebuffer.

xwud -in /usr/tmp/Xvfb_screen0

Displays screen 0 of the server started by the preceding example.

SEE ALSO

X(7), Xserver(1), xwd(1), xwud(1), XWDFile.h

AUTHORS

David P. Wiggins, The Open Group, Inc.

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(7)*.

-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.

-silent Operate silently, i.e. don't ring any bells before and after dumping the window.

ENVIRONMENT

DISPLAY
To get default host and display number.

FILES

XWDFile.h
X Window Dump File format definition file.

SEE ALSO

xwud(1), X(7)

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(7)*.

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(7), 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.

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>] [−scale] [−help] [−rv] [−plane *number*] [−fg *color*] [−bg *color*] [−dumpheader]

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(7)*.

−dumpheader
This option prints out the XWD header information only. Nothing is displayed.

−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.

−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.

–scale Allow the window to be resized, and scale the image to the size of the window.

–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.

BUGS

xwud doesn't handle big/deep images very well on servers that don't have the BIG-REQUESTS extension.

SEE ALSO

xwd(1), xstdcmap(1), X(7)

AUTHOR

Bob Scheifler, MIT X Consortium

NAME

PolyglotMan, rman - reverse compile man pages from formatted form to a number of source formats

SYNOPSIS

rman [*options*] [*file*]

DESCRIPTION

PolyglotMan takes man pages from most of the popular flavors of UNIX and transforms them into any of a number of text source formats. PolyglotMan was formerly known as RosettaMan. The name of the binary is still called *rman* , for scripts that depend on that name; mnemonically, just think "reverse man". Previously *PolyglotMan* required pages to be formatted by nroff prior to its processing. With version 3.0, it *prefers [tn]roff source* and usually produces results that are better yet. And source processing is the only way to translate tables. Source format translation is not as mature as formatted, however, so try formatted translation as a backup.

In parsing [tn]roff source, one could implement an arbitrarily large subset of [tn]roff, which I did not and will not do, so the results can be off. I did implement a significant subset of those use in man pages, however, including tbl (but not eqn), if tests, and general macro definitions, so usually the results look great. If they don't, format the page with nroff before sending it to PolyglotMan. If PolyglotMan doesn't recognize a key macro used by a large class of pages, however, e-mail me the source and a uuencoded nroff-formatted page and I'll see what I can do. When running PolyglotMan with man page source that includes or redirects to other [tn]roff source using the .so (source or inclusion) macro, you should be in the parent directory of the page, since pages are written with this assumption. For example, if you are translating /usr/man/man1/ls.1, first cd into /usr/man.

PolyglotMan accepts man pages from: SunOS, Sun Solaris, Hewlett-Packard HP-UX, AT&T System V, OSF/1 aka Digital UNIX, DEC Ultrix, SGI IRIX, Linux, FreeBSD, SCO. Source processing works for: SunOS, Sun Solaris, Hewlett-Packard HP-UX, AT&T System V, OSF/1 aka Digital UNIX, DEC Ultrix. It can produce printable ASCII-only (control characters stripped), section headers-only, Tk, TkMan, [tn]roff (traditional man page source), SGML, HTML, MIME, LaTeX, LaTeX2e, RTF, Perl 5 POD. A modular architecture permits easy addition of additional output formats.

The latest version of PolyglotMan is always available from *ftp://ftp.cs.berkeley.edu/ucb/people/phelps/tcltk/rman.tar.Z* .

OPTIONS

The following options should not be used with any others and exit PolyglotMan without processing any input.

-h|--help
: Show list of command line options and exit.

-v|--version
: Show version number and exit.

You should specify the filter first, as this sets a number of parameters, and then specify other options.

-f|--filter <ASCII|roff|TkMan|Tk|Sections|HTML|SGML|MIME|LaTeX|LaTeX2e|RTF|POD>
: Set the output filter. Defaults to ASCII.

-S|--source
: PolyglotMan tries to automatically determine whether its input is source or formatted; use this option to declare source input.

-F|--format|--formatted
: PolyglotMan tries to automatically determine whether its input is source or formatted; use this option to declare formatted input.

-l|--title *printf-string*
: In HTML mode this sets the <TITLE> of the man pages, given the same parameters as *-r* .

-r|--reference|--manref *printf-string*
: In HTML and SGML modes this sets the URL form by which to retrieve other man pages. The string can use two supplied parameters: the man page name and its section. (See the Examples section.) If the string is null (as if set from a shell by "-r ''"), '-' or

'off', then man page references will not be HREFs, just set in italics. If your printf supports XPG3 positions specifier, this can be quite flexible.

-V|--volumes *<colon-separated list>*
: Set the list of valid volumes to check against when looking for cross-references to other man pages. Defaults to *1:2:3:4:5:6:7:8:9:o:l:n:p* (volume names can be multicharacter). If an non-whitespace string in the page is immediately followed by a left parenthesis, then one of the valid volumes, and ends with optional other characters and then a right parenthesis--then that string is reported as a reference to another manual page. If this -V string starts with an equals sign, then no optional characters are allowed between the match to the list of valids and the right parenthesis. (This option is needed for SCO UNIX.)

The following options apply only when formatted pages are given as input. They do not apply or are always handled correctly with the source.

-b|--subsections
: Try to recognize subsection titles in addition to section titles. This can cause problems on some UNIX flavors.

-K|--nobreak
: Indicate manual pages don't have page breaks, so don't look for footers and headers around them. (Older nroff -man macros always put in page breaks, but lately some vendors have realized that printout are made through troff, whereas nroff -man is used to format pages for reading on screen, and so have eliminated page breaks.) *PolyglotMan* usually gets this right even without this flag.

-k|--keep
: Keep headers and footers, as a canonical report at the end of the page. changeleft Move changebars, such as those found in the Tcl/Tk manual pages, to the left. --> notaggressive *Disable* aggressive man page parsing. Aggressive manual, which is on by default, page parsing elides headers and footers, identifies sections and more. -->

-n|--name *name*
: Set name of man page (used in roff format). If the filename is given in the form " *name* . *section* ", the name and section are automatically determined. If the page is being parsed from [tn]roff source and it has a .TH line, this information is extracted from that line.

-p|--paragraph
: paragraph mode toggle. The filter determines whether lines should be linebroken as they were by nroff, or whether lines should be flowed together into paragraphs. Mainly for internal use.

-s|section #
: Set volume (aka section) number of man page (used in roff format). tables Turn on aggressive table parsing. -->

-t|--tabstops #
: For those macros sets that use tabs in place of spaces where possible in order to reduce the number of characters used, set tabstops every # columns. Defaults to 8.

NOTES ON FILTER TYPES

ROFF

Some flavors of UNIX ship man page without [tn]roff source, making one's laser printer little more than a laser-powered daisy wheel. This filer tries to intuit the original [tn]roff directives, which can then be recompiled by [tn]roff.

TkMan

TkMan, a hypertext man page browser, uses *PolyglotMan* to show man pages without the (usually) useless headers and footers on each pages. It also collects section and (optionally) subsection heads for direct access from a pulldown menu. TkMan and Tcl/Tk, the toolkit in which it's written, are available via anonymous ftp from *ftp://ftp.smli.com/pub/tcl/*

Tk

This option outputs the text in a series of Tcl lists consisting of text-tags pairs, where tag names roughly correspond to HTML. This output can be inserted into a Tk text widget by doing an *eval <textwidget> insert end <text>* . This format should be relatively easily parsable by other programs that want both the text and the tags. Also see ASCII.

ASCII

When printed on a line printer, man pages try to produce special text effects by overstriking characters with themselves (to produce bold) and underscores (underlining). Other text processing software, such as text editors, searchers, and indexers, must counteract this. The ASCII filter strips away this formatting. Piping nroff output through *col -b* also strips away this formatting, but it leaves behind unsightly page headers and footers. Also see Tk.

Sections

Dumps section and (optionally) subsection titles. This might be useful for another program that processes man pages.

HTML

With a simple extension to an HTTP server for Mosaic or other World Wide Web browser, *PolyglotMan* can produce high quality HTML on the fly. Several such extensions and pointers to several others are included in *PolyglotMan* 's *contrib* directory.

SGML

This is approaching the Docbook DTD, but I'm hoping that someone that someone with a real interest in this will polish the tags generated. Try it to see how close the tags are now.

MIME

MIME (Multipurpose Internet Mail Extensions) as defined by RFC 1563, good for consumption by MIME-aware e-mailers or as Emacs (>=19.29) enriched documents.

LaTeX and LaTeX2e

Why not?

RTF

Use output on Mac or NeXT or whatever. Maybe take random man pages and integrate with NeXT's documentation system better. Maybe NeXT has own man page macros that do this.

PostScript and FrameMaker

To produce PostScript, use *groff* or *psroff* . To produce FrameMaker MIF, use FrameMaker's builtin filter. In both cases you need *[tn]roff* source, so if you only have a formatted version of the manual page, use *PolyglotMan* 's roff filter first.

EXAMPLES

To convert the *formatted* man page named *ls.1* back into [tn]roff source form:

rman -f roff /usr/local/man/cat1/ls.1 > /usr/local/man/man1/ls.1

Long man pages are often compressed to conserve space (compression is especially effective on formatted man pages as many of the characters are spaces). As it is a long man page, it probably has subsections, which we try to separate out (some macro sets don't distinguish subsections well enough for *PolyglotMan* to detect them). Let's convert this to LaTeX format:

pcat /usr/catman/a_man/cat1/automount.z | rman -b -n automount -s 1 -f latex > automount.man

Alternatively, *man 1 automount | rman -b -n automount -s 1 -f latex > automount.man*

For HTML/Mosaic users, *PolyglotMan* can, without modification of the source code, produce HTML links that point to other HTML man pages either pregenerated or generated on the fly. First let's assume pregenerated HTML versions of man pages stored in */usr/man/html* . Generate these one-by-one with the following form:
rman -f html -r 'http:/usr/man/html/%s.%s.html' /usr/man/cat1/ls.1 > /usr/man/html/ls.1.html

If you've extended your HTML client to generate HTML on the fly you should use something like:
rman -f html -r 'http:~/bin/man2html?%s:%s' /usr/man/cat1/ls.1
when generating HTML.

BUGS/INCOMPATIBILITIES

PolyglotMan is not perfect in all cases, but it usually does a good job, and in any case reduces the problem of converting man pages to light editing.

Tables in formatted pages, especially H-P's, aren't handled very well. Be sure to pass in source for the page to recognize tables.

The man pager *woman* applies its own idea of formatting for man pages, which can confuse *PolyglotMan* . Bypass *woman* by passing the formatted manual page text directly into *PolyglotMan* .

The [tn]roff output format uses fB to turn on boldface. If your macro set requires .B, you'll have to a postprocess the *PolyglotMan* output.

SEE ALSO

tkman(1) , *xman(1)* , *man(1)* , *man(7)* or *man(5)* depending on your flavor of UNIX

AUTHOR

PolyglotMan
by Thomas A. Phelps (*phelps@ACM.org*)
developed at the
University of California, Berkeley
Computer Science Division

Manual page last updated on $Date: 1998/07/13 09:47:28 $