NAME

xdm – X Display Manager with support for XDMCP

SYNOPSIS

xdm [**–config** *configuration_file*] [**–nodaemon**] [**–debug** *debug_level*] [**–error** *error_log_file*] [**–resources** *resource_file*] [**–server** *server_entry*] [**–session** *session_program*]

DESCRIPTION

Xdm manages a collection of X displays, which may be on the local host or remote servers. The design of **xdm** was guided by the needs of X terminals as well as the X Consortium standard XDMCP, the *X Display Manager Control Protocol*. **Xdm** provides services similar to those provided by **init**(1M)**, getty**(1M), and **login**(1) on character terminals: prompting for login name and password, authenticating the user, and running a ''session.''

A ''session'' is defined by the lifetime of a particular process; in the traditional character-based terminal world, it is the user's login shell. In the **xdm** context, it is an arbitrary session manager. This is because in a windowing environment, a user's login shell process does not necessarily have any terminal-like interface with which to connect. When a real session manager is not available, a window manager or terminal emulator is typically used as the ''session manager,'' meaning that termination of this process terminates the user's session.

When the session is terminated, **xdm** resets the X server and (optionally) restarts the whole process.

When **xdm** receives an Indirect query via XDMCP, it can run a chooser **process to perform an XDMCP BroadcastQuery (or an XDMCP Query to specified hosts) on behalf of the display and offer a menu of possible hosts that offer XDMCP display management. This feature is useful with X terminals that do not offer a host menu themselves.**

Because **xdm** provides the first interface that users will see, it is designed to be simple to use and easy to customize to the needs of a particular site. **Xdm** has many options, most of which have reasonable defaults. Browse through the various sections of this manual, picking and choosing the things you want to change. Pay particular attention to the **Session Program** section, which will describe how to set up the style of session desired.

TYPICAL USAGE

Actually, **xdm** is designed to operate in such a wide variety of environments that *typical* is probably a misnomer.

First, the **xdm** configuration file should be set up. Make a directory (usually */usr/openwin/lib/xdm*) to contain all of the relevant files. Here is a reasonable configuration file, which could be named *xdm-config*:

```
DisplayManager.servers:          /usr/openwin/lib/xdm/Xservers
DisplayManager.errorLogFile:     /usr/tmp/xdm-errors
DisplayManager*resources:        /usr/openwin/lib/xdm/Xresources
DisplayManager*startup:          /usr/openwin/lib/xdm/Xstartup
```

```
DisplayManager*session:          /usr/openwin/lib/xdm/Xsession
DisplayManager.pidFile:          /usr/tmp/xdm-pid
DisplayManager._0.authorize:     true
DisplayManager*authorize:        false
```

Note that this file simply contains references to other files. Note also that some of the resources are specified with ‘‘*’’ separating the components. These resources can be made unique for each different display, by replacing the ‘‘*’’ with the display-name, but normally this is not very useful. See the **Resources** section for a complete discussion.

The first file, */usr/openwin/lib/xdm/Xservers*, contains the list of displays to manage that are not using XDMCP. Most workstations have only one display, numbered 0, so the file will look something like this:

```
:0 Local local /usr/openwin/bin/X :0
```

This will keep */usr/openwin/bin/X* running on this display and manage a continuous cycle of sessions.

The file */usr/tmp/xdm-errors* will contain error messages from **xdm** and anything output to stderr by *Xsetup*, *Xstartup*, *Xsession* or *Xreset*. When you have trouble getting **xdm** working, check this file to see if **xdm** has any clues to the trouble.

The next configuration entry, */usr/openwin/lib/xdm/Xresources*, is loaded onto the display as a resource database using **xrdb**(1). As the authentication widget reads this database before starting up, it usually contains parameters for that widget:

```
xlogin*login.translations: #override\
        <Key>F1: set-session-argument(failsafe) finish-field()\n\
        <Key>Return: set-session-argument() finish-field()
xlogin*borderWidth: 3
#ifdef COLOR
xlogin*greetColor: CadetBlue
xlogin*failColor: red
#endif
```

Please note the translations entry; it specifies a few new translations for the widget which allow users to escape from the default session (and avoid troubles that may occur in it). Note that if #override is not specified, the default translations are removed and replaced by the new value, not a very useful result as some of the default translations are quite useful (such as ‘‘<Key>: insert-char ()’’ which responds to normal typing).

The *Xstartup* file shown here simply prevents login while the file */etc/nologin* exists. As there is no provision for displaying any messages here (there isn’t any core X client which displays files), the user will probably be baffled by this behavior. Thus this is not a complete example, but simply a demonstration of the available functionality.

Here is a sample *Xstartup* script:

```
#!/bin/sh
#
# Xstartup
#
# This program is run as root after the user is verified
#
if [ –f /etc/nologin ]; then
        exit 1
fi
exit 0
```

The most interesting script is *Xsession*. This version recognizes the special ''failsafe'' mode, specified in the translations in the *Xresources* file above, to provide an escape from the ordinary session:

```
#!/bin/sh
#
# Xsession
#
# This is the program that is run as the client
# for the display manager. This example is
# quite friendly as it attempts to run a per-user
# .xsession file instead of forcing a particular
# session layout
#

case $# in
1)
        case $1 in
        failsafe)
                exec xterm –geometry 80x24–0–0 –ls
                ;;
        esac
esac

startup=$HOME/.xsession
resources=$HOME/.Xresources

if [ –f $startup ]; then
        exec $startup
        exec /bin/sh $startup
else
        if [ ! –f $resources ]; then
                resources=$HOME/.Xdefaults
```

```
                fi
                if [ –f $resources ]; then
                        xrdb –load $resources
                fi
                twm &
                exec xterm –geometry 80x24+10+10 –ls
        fi
```

OPTIONS

All of these options, except –**config**, specify values which 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/openwin/lib/xdm/xdm-config* is the default.

–**nodaemon**
: Specifies ''false'' as the value for the **DisplayManager.daemonMode** resource. This suppresses the normal daemon behavior, which is for **xdm** to close all file descriptors, disassociate itself from the controlling terminal, and put itself in the background when it first starts up.

–**debug** *debug_level*
: Specifies the numeric value for the **DisplayManager.debugLevel** resource. A non-zero value causes **xdm** to print lots of debugging statements to the terminal; it also disables the **DisplayManager.daemonMode** resource, forcing **xdm** to run synchronously. To interpret these debugging messages, a copy of the source code for **xdm** is almost a necessity. No attempt has been made to rationalize or standardize the output.

–**error** *error_log_file*
: Specifies the value for the **DisplayManager.errorLogFile** resource. This file contains errors from **xdm** as well as anything written to stderr by the various scripts and programs run during the progress of the session.

–**resources** *resource_file*
: Specifies the value for the **DisplayManager*resources** resource. This file is loaded using **xrdb**(1) to specify configuration parameters for the authentication widget.

–**server** *server_entry*
: Specifies the value for the **DisplayManager.servers** resource. See the section **Server Specification** for a description of this resource.

–**udpPort** *port_number*
: Specifies the value for the **DisplayManager.requestPort** resource. This sets the port-number which **xdm** will monitor for XDMCP requests. As XDMCP uses the registered well-known UDP port 177, this resource should not be changed except for debugging.

–*session session_program*
: Specifies the value for the **DisplayManager*session** resource. This indicates the

program to run as the session after the user has logged in.

–xrm resource_specification
Allows an arbitrary resource to be specified, as in most X Toolkit applications.

RESOURCES

At many stages the actions of **xdm** can be controlled through the use of its configuration file, which is in the X resource format. Some resources modify the behavior of **xdm** on all displays, while others modify its behavior on a single display. Where actions relate to a specific display, the display name is inserted into the resource name between ''Display-Manager'' and the final resource name segment. For example, **DisplayManager.expo_0.startup** is the name of the resource which defines the startup shell file on the ''expo:0'' display. Because the resource manager uses colons to separate the name of the resource from its value and dots to separate resource name parts, **xdm** substitutes underscores for both dots and colons when generating the resource name.

DisplayManager.servers
This resource either specifies a file name full of server entries, one per line (if the value starts with a slash), or a single server entry. See the section **Server Specification** for the details.

DisplayManager.requestPort
This indicates the UDP port number which **xdm** uses to listen for incoming XDMCP requests. Unless you need to debug the system, leave this with its default value of 177.

DisplayManager.errorLogFile
Error output is normally directed at the system console. To redirect it, set this resource to a file name. A method to send these messages to *syslog* should be developed for systems which support it; however, the wide variety of interfaces precludes any system-independent implementation. This file also contains any output directed to stderr by the *Xsetup*, *Xstartup*, *Xsession* and *Xreset* files, so it will contain descriptions of problems in those scripts as well.

DisplayManager.debugLevel
If the integer value of this resource is greater than zero, reams of debugging information will be printed. It also disables daemon mode, which would redirect the information into the bit-bucket, and allows non-root users to run **xdm,** which would normally not be useful.

DisplayManager.daemonMode
Normally, **xdm** attempts to make itself into a daemon process unassociated with any terminal. This is accomplished by forking and leaving the parent process to exit, then closing file descriptors and releasing the controlling terminal. In some environments this is not desired (in particular, when debugging). Setting this resource to ''false'' will disable this feature.

DisplayManager.pidFile
The filename specified will be created to contain an ASCII representation of the process-id of the main **xdm** process. **Xdm** also uses file locking on this file to attempt to eliminate multiple daemons running on the same machine, which

would cause quite a bit of havoc.

DisplayManager.lockPidFile

This is the resource which controls whether **xdm** uses file locking to keep multiple display managers from running amok. On System V, this uses the **lockf**(3C) library call, while on BSD it uses **flock**(3B)

DisplayManager.remoteAuthDir

This names a directory in which **xdm** stores authorization files while initializing the session. The default value is */usr/openwin/lib/xdm.*

DisplayManager.autoRescan

This boolean controls whether **xdm** rescans the configuration, servers, access control and authentication keys files after a session terminates and the files have changed. By default it is ''true.'' You can force **xdm** to reread these files by sending a SIGHUP to the main process.

DisplayManager.removeDomainname

When computing the display name for XDMCP clients, the name resolver will typically create a fully qualified host name for the terminal. As this is sometimes confusing, **xdm** will remove the domain name portion of the host name if it is the same as the domain name of the local host when this variable is set. By default the value is ''true.''

DisplayManager.keyFile

XDM-AUTHENTICATION-1 style XDMCP authentication requires that a private key be shared between **xdm** and the terminal. This resource specifies the file containing those values. Each entry in the file consists of a display name and the shared key. By default, **xdm** does not include support for XDM-AUTHENTICATION-1, as it requires DES which is not generally distributable because of United States export restrictions.

DisplayManager.accessFile

To prevent unauthorized XDMCP service and to allow forwarding of XDMCP IndirectQuery requests, this file contains a database of hostnames which are either allowed direct access to this machine, or have a list of hosts to which queries should be forwarded to. The format of this file is described in the section **XDMCP Access Control.**

DisplayManager.DISPLAY.resources

This resource specifies the name of the file to be loaded by **xrdb**(3) 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/openwin/lib/xdm/Xresources* is the conventional name.

DisplayManager.DISPLAY.chooser

Specifies the program run to offer a host menu for Indirect queries redirected to the special host name CHOOSER. */usr/openwin/lib/xdm/chooser* is the default. See the sections **XDMCP Access Control** and **Chooser**.

DisplayManager.DISPLAY.xrdb
Specifies the program used to load the resources. By default, **xdm** uses */usr/openwin/bin/xrdb*.

DisplayManager.DISPLAY.cpp
This specifies the name of the C preprocessor which is used by **xrdb**(1).

DisplayManager.DISPLAY.setup
This specifies a program which is run (as root) before offering the Login window. This may be used to change the appearence of the screen around the Login window or to put up other windows (e.g., you may want to run *xconsole* here). By default, no program is run. The conventional name for a file used here is *Xsetup*. See the section **Setup Program.**

DisplayManager.DISPLAY.startup
This specifies a program which is run (as root) after the authentication process succeeds. By default, no program is run. The conventional name for a file used here is *Xstartup*. See the section **Startup Program.**

DisplayManager.DISPLAY.session
This specifies the session to be executed (not running as root). By default, */usr/openwin/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. Again, by default no program is run. The conventional name is *Xreset*. See the section **Reset Program.**

DisplayManager.DISPLAY.openDelay

DisplayManager.DISPLAY.openRepeat

DisplayManager.DISPLAY.openTimeout

DisplayManager.DISPLAY.startAttempts
These numeric resources control the behavior of **xdm** when attempting to open intransigent servers. **openDelay** is the length of the pause (in seconds) between successive attempts, **openRepeat** is the number of attempts to make, **openTimeout** is the amount of time to wait while actually attempting the open (i.e., the maximum time spent in the *connect*(2) system call) and **startAttempts** is the number of times this entire process is done before giving up on the server. After **openRepeat** attempts have been made, or if **openTimeout** seconds elapse in any particular attempt, **xdm** terminates and restarts the server, attempting to connect again. This process is repeated **startAttempts** times, at which point the display is declared dead and disabled. Although this behavior may seem arbitrary, it has been empirically developed and works quite well on most systems. The default values are 5 for **openDelay**, 5 for **openRepeat**, 30 for **openTimeout** and 4 for **startAttempts**.

DisplayManager.DISPLAY.pingInterval

DisplayManager.DISPLAY.pingTimeout

To discover when remote displays disappear, **xdm** occasionally pings them, using an X connection and *XSync* calls. **pingInterval** specifies the time (in minutes) between each ping attempt, **pingTimeout** specifies the maximum amount of time (in minutes) to wait for the terminal to respond to the request. If the terminal does not respond, the session is declared dead and terminated. By default, both are set to 5 minutes. If you frequently use X terminals which can become isolated from the managing host, you may wish to increase this value. The only worry is that sessions will continue to exist after the terminal has been accidentally disabled. **xdm** will not ping local displays. Although it would seem harmless, it is unpleasant when the workstation session is terminated as a result of the server hanging for NFS service and not responding to the ping.

DisplayManager.DISPLAY.terminateServer

This boolean resource specifies whether the X server should be terminated when a session terminates (instead of resetting it). This option can be used when the server tends to grow without bound over time, in order to limit the amount of time the server is run. The default value is ''false.''

DisplayManager.DISPLAY.userPath

Xdm sets the PATH environment variable for the session to this value. It should be a colon separated list of directories; see **sh**(1) for a full description. '':/bin:/usr/bin:/usr/openwin/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/openwin/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/openwin/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** specifies the type to use. Currently, **xdm** supports only MIT-MAGIC-COOKIE-1 authorization. XDM-AUTHORIZATION-1 could be supported as well, but DES is not generally distributable. XDMCP connections specify which authorization types are supported dynamically, so **authName** is ignored in this case. When **authorize** is set for a display and authorization is not available, the user is informed by having a different message displayed in the login widget. By default, **authorize** is ‘‘true’’; **authName** is ‘‘MIT-MAGIC-COOKIE-1.’’

DisplayManager.DISPLAY.authFile
This file is used to communicate the authorization data from **xdm** to the server, using the –**auth** server command line option. It should be kept in a directory which is not world-writable as it could easily be removed, disabling the authorization mechanism in the server.

DisplayManager.DISPLAY.resetForAuth
The original implementation of authorization in the sample server reread the authorization file at server reset time, instead of when checking the initial connection. As **xdm** generates the authorization information just before connecting to the display, an old server would not get up-to-date authorization information. This resource causes **xdm** to send SIGHUP to the server after setting up the file, causing an additional server reset to occur, during which time the new authorization information will be read. The default is ‘‘false,’’ which will work for all MIT servers.

DisplayManager.DISPLAY.userAuthDir
When **xdm** is unable to write to the usual user authorization file ($HOME/.Xauthority), it creates a unique file name in this directory and points the environment variable XAUTHORITY at the created file. It uses */tmp* by default.

XDMCP ACCESS CONTROL

The database file specified by the **DisplayManager.accessFile** provides information which **xdm** uses to control access from displays requesting XDMCP service. This file contains three types of entries: entries which control the response to Direct and Broadcast queries, entries which control the response to Indirect queries, and macro definitions.

The format of the Direct entries is simple, either a host name or a pattern, which is distinguished from a host name by the inclusion of one or more meta characters ('∗' matches any sequence of 0 or more characters, and '?' matches any single character) which are compared against the host name of the display device. If the entry is a host name, all comparisons are done using network addresses, so any name which converts to the correct network address may be used. For patterns, only canonical host names are used in the comparison, so ensure that you do not attempt to match aliases. Preceding either a host name or a pattern with a '!' character causes hosts which match that entry to be excluded.

An Indirect entry also contains a host name or pattern, but follows it with a list of host names or macros to which indirect queries should be sent.

A macro definition contains a macro name and a list of host names and other macros that the macro expands to. To distinguish macros from hostnames, macro names start with a '%' character. Macros may be nested.

Indirect entries may also specify to have **xdm** run *chooser* to offer a menu of hosts to connect to. See the section **Chooser**.

When checking access for a particular display host, each entry is scanned in turn and the first matching entry determines the response. Direct and Broadcast entries are ignored when scanning for an Indirect entry and vice-versa.

Blank lines are ignored, '#' is treated as a comment delimiter causing the rest of that line to be ignored, and '*newline*' causes the newline to be ignored, allowing indirect host lists to span multiple lines.

Here is an example Xaccess file:

```
#
# Xaccess – XDMCP access control file
#

#
# Direct/Broadcast query entries
#

!xtra.lcs.mit.edu          # disallow direct/broadcast service for xtra
bambi.ogi.edu              # allow access from this particular display
∗.lcs.mit.edu              # allow access from any display in LCS

#
# Indirect query entries
#

%HOSTS                     expo.lcs.mit.edu xenon.lcs.mit.edu \
                           excess.lcs.mit.edu kanga.lcs.mit.edu

extract.lcs.mit.edu        xenon.lcs.mit.edu          #force extract to contact xenon
```

!xtra.lcs.mit.edu	dummy	#disallow indirect access
∗.lcs.mit.edu	%HOSTS	#all others get to choose

CHOOSER

For X terminals that do not offer a host menu for use with Broadcast or Indirect queries, the *chooser* program can do this for them. In the *Xaccess* file, specify ''CHOOSER'' as the first entry in the Indirect host list. *Chooser* will send a Query request to each of the remaining host names in the list and offer a menu of all the hosts that respond.

The list may consist of the word ''BROADCAST,'' in which case *chooser* will send a Broadcast instead, again offering a menu of all hosts that respond. Note that on some operating systems, UDP packets cannot be broadcast, so this feature will not work.

Example *Xaccess* file using *chooser*:

extract.lcs.mit.edu	CHOOSER %HOSTS	#offer a menu of these hosts
xtra.lcs.mit.edu	CHOOSER BROADCAST	#offer a menu of all hosts

The program to use for **chooser** is specified by the **DisplayManager.DISPLAY.chooser** resource. Resources for this program can be put into the file named by **DisplayManager.DISPLAY.resources**.

SERVER SPECIFICATION

The resource **DisplayManager.servers** gives a server specification or, if the values starts with a slash (/), the name of a file containing server specifications, one per line.

Each specification indicates a display which should constantly be managed and which is not using XDMCP. Each consists of at least three parts: a display name, a display class, a display type, and (for local servers) a command line to start the server. A typical entry for local display number 0 would be:

```
:0 local /usr/openwin/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 local /usr/openwin/bin/X :0'' instead of ''localhost:0 local /usr/openwin/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 (like a corral of X terminals) and would like to set resources for groups of them. When using XDMCP, the display is required to specify the display class, so the manual for your particular X terminal should document the display class string for your device. If it doesn't, you can run **xdm** in debug mode and look at the resource strings which it generates for that device, which will include the class string.

SETUP PROGRAM

The *Xsetup* file is run after the server is reset, but before the Login window is offered. The file is typically a shell script. It is run as root, so should be careful about security. This is the place to change the root background or bring up other windows that should appear on the screen along with the Login widget. Note that since **xdm** grabs the keyboard, any other windows will not be able to receive keyboard input. They will be able to interact with the mouse, however; beware of potential security holes here. If **DisplayManager.DISPLAY.grabServer** is set, *Xsetup* will not be able to connect to the display at all. Resources for this program can be put into the file named by **DisplayManager.DISPLAY.resources**.

AUTHENTICATION WIDGET

The authentication widget reads a name/password pair from the keyboard. Nearly every imaginable parameter can be controlled with a resource. Resources for this widget should be put into the file named by **DisplayManager.DISPLAY.resources**. All of these have reasonable default values, so it is not necessary to specify any of them.

xlogin.Login.width, xlogin.Login.height, xlogin.Login.x, xlogin.Login.y
The geometry of the Login widget is normally computed automatically. If you wish to position it elsewhere, specify each of these resources.

xlogin.Login.foreground
The color used to display the typed-in user name.

xlogin.Login.font
The font used to display the typed-in user name.

xlogin.Login.greeting
A string which identifies this window. The default is ‘‘X Window System.’’

xlogin.Login.unsecureGreeting
When X authorization is requested in the configuration file for this display and none is in use, this greeting replaces the standard greeting. The default is ‘‘This is an unsecure session’’

xlogin.Login.greetFont
The font used to display the greeting.

xlogin.Login.greetColor
The color used to display the greeting.

xlogin.Login.namePrompt
The string displayed to prompt for a user name. **Xrdb** strips trailing white space from resource values, so to add spaces at the end of the prompt (usually a nice thing), add spaces escaped with backslashes. The default is ‘‘Login: ’’

xlogin.Login.passwdPrompt
The string displayed to prompt for a password. The default is ‘‘Password: ’’

xlogin.Login.promptFont
The font used to display both prompts.

xlogin.Login.promptColor
The color used to display both prompts.

xlogin.Login.fail

A message which is displayed when the authentication fails. The default is ''Login incorrect''

xlogin.Login.failFont
The font used to display the failure message.

xlogin.Login.failColor
The color used to display the failure message.

xlogin.Login.failTimeout
The number of seconds that the failure message is displayed. The default is 30.

xlogin.Login.translations
This specifies the translations used for the login widget. Refer to the X Toolkit documentation for a complete discussion on translations. The default translation table is:

```
Ctrl<Key>H:          delete-previous-character() \n\
Ctrl<Key>D:          delete-character() \n\
Ctrl<Key>B:          move-backward-character() \n\
Ctrl<Key>F:          move-forward-character() \n\
Ctrl<Key>A:          move-to-begining() \n\
Ctrl<Key>E:          move-to-end() \n\
Ctrl<Key>K:          erase-to-end-of-line() \n\
Ctrl<Key>U:          erase-line() \n\
Ctrl<Key>X:          erase-line() \n\
Ctrl<Key>C:          restart-session() \n\
Ctrl<Key>\\:         abort-session() \n\
<Key>BackSpace:      delete-previous-character() \n\
<Key>Delete:         delete-previous-character() \n\
<Key>Return:         finish-field() \n\
<Key>:               insert-char() \
```

The actions which are supported by the widget are:

delete-previous-character
Erases the character before the cursor.

delete-character
Erases the character after the cursor.

move-backward-character
Moves the cursor backward.

move-forward-character
Moves the cursor forward.

move-to-begining
(Apologies about the spelling error.) Moves the cursor to the beginning of the editable text.

move-to-end

Moves the cursor to the end of the editable text.

erase-to-end-of-line
: Erases all text after the cursor.

erase-line
: Erases the entire text.

finish-field
: If the cursor is in the name field, proceeds to the password field; if the cursor is in the password field, checks the current name/password pair. If the name/password pair is valid, **xdm** starts the session. Otherwise the failure message is displayed and the user is prompted again.

abort-session
: Terminates and restarts the server.

abort-display
: Terminates the server, disabling it. This is a rash action and is not accessible in the default configuration. It can be used to stop *xdm* when shutting the system down or when using *xdmshell*.

restart-session
: Resets the X server and starts a new session. This can be used when the resources have been changed and you want to test them or when the screen has been overwritten with system messages.

insert-char
: Inserts the character typed.

set-session-argument
: Specifies a single word argument which is passed to the session at startup. See the sections **Session Program** and **Typical Usage**.

allow-all-access
: Disables access control in the server. This can be used when the .Xauthority file cannot be created by **xdm.** Be very careful using this; it might be better to disconnect the machine from the network before doing this.

STARTUP PROGRAM

The *Xstartup* file is typically a shell script. It is run as root and should be very careful about security. This is the place to put commands which add entries to */etc/utmp*, mount users' home directories from file servers, display the message of the day, or abort the session if logins are not allowed. Various environment variables are set for the use of this script:

DISPLAY	the associated display name
HOME	the initial working directory of the user
USER	the user name
PATH	the value of **DisplayManager.DISPLAY.systemPath**
SHELL	the value of **DisplayManager.DISPLAY.systemShell**
XAUTHORITY	may be set to an authority file

No arguments are passed to the script. **Xdm** waits until this script exits before starting the user session. If the exit value of this script is non-zero, **xdm** discontinues the session and starts another authentication cycle.

SESSION PROGRAM

The *Xsession* program is the command which is run as the user's session. It is run with the permissions of the authorized user, and has several environment variables specified:

DISPLAY	the associated display name
HOME	the initial working directory of the user
USER	the user name
PATH	the value of **DisplayManager.DISPLAY.userPath**
SHELL	the user's default shell (from *getpwnam*)
XAUTHORITY	may be set to a non-standard authority file

At most installations, *Xsession* should look in $HOME for a file *.xsession*, which contains commands that each user would like to use as a session. *Xsession* should also implement a system default session if no user-specified session exists. See the section **Typical Usage**.

An argument may be passed to this program from the authentication widget using the 'set-session-argument' action. This can be used to select different styles of session. One good use of this feature is to allow the user to escape from the ordinary session when it fails. This allows users to repair their own *.xsession* if it fails, without requiring administrative intervention. The section **Typical Usage** demonstrates this feature.

RESET PROGRAM

Symmetrical with *Xstartup*, the *Xreset* script is run after the user session has terminated. Run as root, it should contain commands that undo the effects of commands in *Xstartup*, removing entries from */etc/utmp* or unmounting directories from file servers. The environment variables that were passed to *Xstartup* are also passed to *Xreset*.

CONTROLLING THE SERVER

Xdm controls local servers using POSIX signals. SIGHUP is expected to reset the server, closing all client connections and performing other cleanup duties. SIGTERM is expected to terminate the server. If these signals do not perform the expected actions, **xdm** will not perform properly.

To control remote terminals not using XDMCP, **xdm** searches the window hierarchy on the display and uses the protocol request KillClient in an attempt to clean up the terminal for the next session. This may not actually kill all of the clients, as only those which have created windows will be noticed. XDMCP provides a more sure mechanism; when **xdm** closes its initial connection, the session is over and the terminal is required to close all other connections.

CONTROLLING XDM

Xdm responds to two signals: SIGHUP and SIGTERM. When sent a SIGHUP, **xdm** rereads the configuration file, the access control file, and the servers file. For the servers file, it notices if entries have been added or removed. If a new entry has been added, **xdm** starts a session on the associated display. Entries which have been removed are disabled immediately, meaning that any session in progress will be terminated without notice and no new session will be started.

When sent a SIGTERM, **xdm** terminates all sessions in progress and exits. This can be used when shutting down the system.

Xdm attempts to mark its various sub-processes for **ps**(1) by editing the command line argument list in place. Because **xdm** can't allocate additional space for this task, it is useful to start **xdm** with a reasonably long command line (using the full path name should be enough). Each process which is servicing a display is marked *–display*.

OTHER POSSIBILITIES

You can use **xdm** to run a single session at a time, using the 4.3 **init** options or other suitable daemon by specifying the server on the command line:

    xdm –server ":0 SUN-3/60CG4 local /usr/bin/X :0"

Or, you might have a file server and a collection of X terminals. The configuration for this is identical to the sample above, except the *Xservers* file would look like

    extol:0 VISUAL-19 foreign
    exalt:0 NCD-19 foreign
    explode:0 NCR-TOWERVIEW3000 foreign

This directs **xdm** to manage sessions on all three of these terminals. See the section **Controlling Xdm** for a description of using signals to enable and disable these terminals in a manner reminiscent of **init**(1M).

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.**

On the Solaris x86 platform, an attempt to type a login name into **xdm** will cause the keyboard to lock up. The workaround is to use **xdm -nodaemon** instead.

FILES

/usr/openwin/lib/xdm/xdm-config
the default configuration file

/usr/openwin/lib/xdm/Xaccess
the default access file, listing authorized displays

/usr/openwin/lib/xdm/Xservers
the default server file, listing non-XDMCP servers to manage

$(HOME)/.Xauthority user authorization file where *xdm* stores keys for clients to read

/usr/openwin/lib/xdm/chooser
the default chooser

/usr/openwin/bin/xrdb the default resource database loader

/usr/openwin/bin/X the default server

/usr/openwin/bin/xterm the default session program and failsafe client

/usr/openwin/lib/xdm/A<host>–<suffix>

the default place for authorization files

SEE ALSO

X11(7), **xinit**(1), **xauth**(1), and XDMCP

COPYRIGHT

AUTHOR

Keith Packard, MIT X Consortium