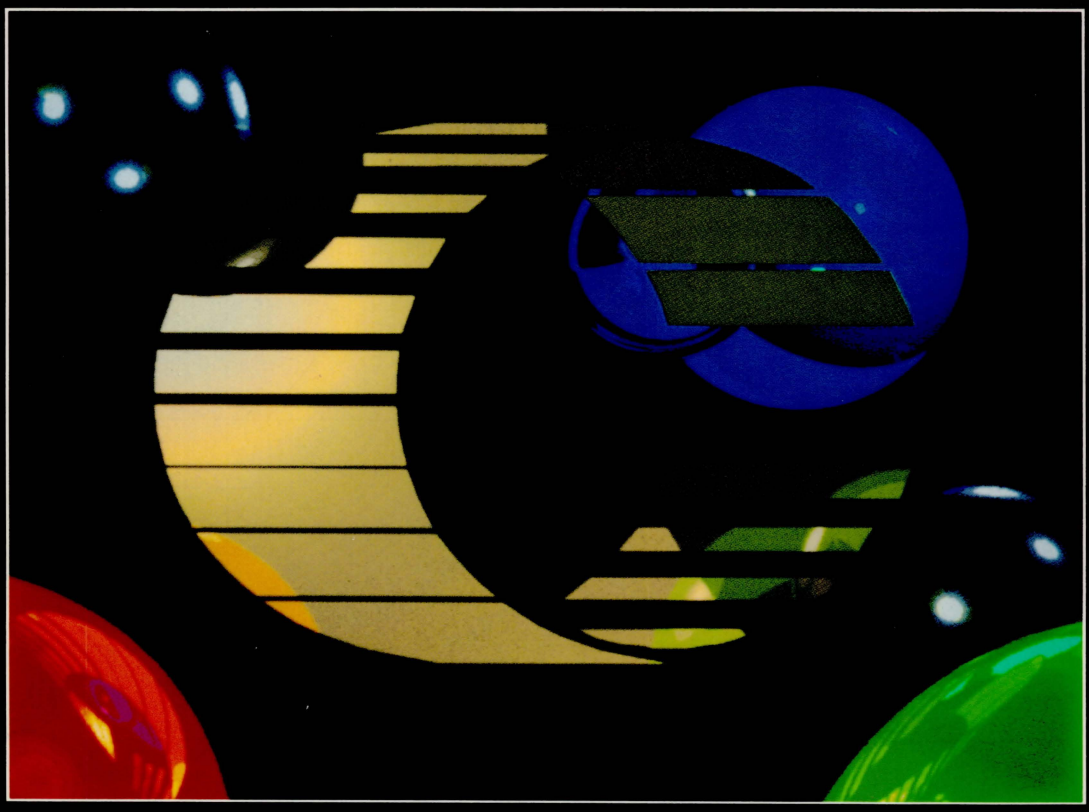
CONVEX OSF/Motif and CXwindows

User's Guide

Preliminary Edition



CONVEX Computer Corporation

Reader Comments

CONVEX OSF/Motif and CXwindows User's Guide

Features

What do you think about the following features and how would you rank them in importance?

	Great			Bad	Ranking; 1-7 (1 is most important)
Page size	☐	☐	☐	☐	_____
Paperback bound	☐	☐	☐	☐	_____
Wire bound	☐	☐	☐	☐	_____
Page layout	☐	☐	☐	☐	_____
Technical information	☐	☐	☐	☐	_____
Manual organization	☐	☐	☐	☐	_____
Examples	☐	☐	☐	☐	_____

Your Suggestions

What changes would improve this guide?

What CONVEX books would you like to see published in this format?

Your Background

Which of the following best describes you?

- ☐ Application programmer
- ☐ Application user
- ☐ System manager
- ☐ System programmer

What Now?

Just fold, tape, and mail.
Thank you for filling in this comment card.

CONVEX

NO POSTAGE
NECESSARY IF
MAILED IN THE
UNITED STATES

Business Reply Mail
FIRST CLASS PERMIT NO. 1046 RICHARDSON, TEXAS

POSTAGE WILL BE PAID BY ADRESSEE

ATTENTION: Neal Johnston
CONVEX Computer Corporation
P.O. Box 833851
Richardson, TX 75083-3851

CONVEX OSF/Motif
and
CXwindows
User's Guide

IMPORTANT

CONVEX approves only limited distribution of this preliminary document. This sample document is provided to select users to solicit feedback.

The CONVEX documentation staff is very interested in your evaluation of this book's usefulness. Please fill in the survey at the front of this book.

The information in this book is preliminary and CONVEX makes no claims of technical accuracy. This pre-release information is subject to change without notice and should not be interpreted as any commitment by CONVEX.

CONVEX OSF/Motif
and
CXwindows
User's Guide
Preliminary Edition

by Neal W. Johnston

CONVEX Computer Corporation

CONVEX OSF/Motif and CXwindows
User's Guide
Preliminary Edition

Printed in the United States of America

Acknowledgments

The author wishes to thank all the people at CONVEX and the users who contributed their ideas and time in the development of this book.

Special thanks to: Kathy Frisch, Marilyn Johnstone, Greg Stiehl, and Martin Streicher.

Cover Designs: F&J Inc.

Cover Artwork: Kent Fuka

About the Cover: The color and transparent spheres along with the CONVEX "C" logo were generated on a C220 using Rayshade V3.0, a ray-tracing program. Rayshade reads an ASCII file that describes a scene to be rendered and produces a Utah Raster RLE format file of the image.

Table of Contents

CONVEX OSF/Motif and CXwindows User's Guide, Preliminary Edition

How to Use This Guide v
- Purpose and Audience v
- Organization v
- Notational Conventions vi
- Associated Documents vii
- Ordering Documentation vii
- Technical Assistance viii
- Reader Comments viii

Product Overviews 1-1
- CXwindows V2.0 1-1
  - Supported Core Clients 1-1
  - Unsupported Clients 1-2
  - CONVEX Clients 1-2
  - Contributed Clients 1-2
  - X Programming Libraries 1-3
  - Examples and Demos 1-3
- CONVEX OSF/Motif V1.0 1-4
  - Supported Core Clients 1-4
  - OSF/Motif Programming Libraries 1-4

The X Window System 2-1
- Background 2-1
- The X Display 2-1
- Interactions 2-2
- The Server-Client Model 2-3
- Clients 2-4
  - The Window Manager 2-4
  - The Terminal Emulator 2-4
  - The Display Manager 2-5
  - Other Clients 2-5

X Command Line Options 3-1
- Common Options 3-1
- Specifying Fonts 3-1
- Specifying Other Options 3-2

X Application Defaults 4-1
Resources and Preferences 4-1
Specifying Resources 4-1
Instances and Classes 4-2
Specifying Resources for the xedit Client 4-3
Specific Resources for xedit 4-5
Writing Resources for Other X Clients 4-5
Precedence Rules for Resources 4-6
Using xrdb to Configure the X Server 4-6
Setting Resources 4-7
Example Scenario 4-7

Window Managers 5-1
The Current View 5-1
twm 5-1
Variables 5-2
Buttons 5-3
Menus 5-4
mwm 5-6

Configuring the X Display Manager 6-1
Control Protocol 6-1
The X Display Manager 6-1
xdm Configuration and Setup Files 6-2
xdm-config 6-2
Xstartup 6-4
Xsession 6-5
Xreset 6-6
Running xdm Servers with XDMCP 6-7
Adding New Servers 6-7
Interrupting xdm 6-8
Non-XDMCP Servers and the Xservers File 6-8
Typical xdm Errors 6-9
Invoking xdm 6-10

Font Management 7-1
Specifying Fonts 7-1
Defining fonts.dir Files 7-1
Adding Fonts 7-2
Making Font Aliases 7-2
Making the Server Aware of Aliases 7-3
What Now? 7-3

Error Messages A-1
For CXwindows A-1
For CONVEX OSF/Motif A-2

Glossary Glossary-1

How to Use This Guide

Purpose and Audience

The *CONVEX OSF/Motif and CXwindows User's Guide* provides information on the X Window System from M.I.T. and OSF/Motif from the Open Software Foundation, discusses how CONVEX has implemented these products, and details how to configure and begin using them.

This guide addresses:

- ❑ Users who will run applications under CONVEX OSF/Motif and CXwindows.
- ❑ System managers who will configure and maintain CONVEX OSF/Motif and CXwindows.

Organization

This guide is organized in the following manner:

- ❑ Chapter 1, "Product Overviews," presents overviews of the CXwindows V2.0 and CONVEX OSF/Motif V1.0 products.
- ❑ Chapter 2, "The X Window System," describes basic concepts and information about X and how it works.
- ❑ Chapter 3, "X Command Line Options," contains information on the common set of command line options X applications share.
- ❑ Chapter 4, "X Application Defaults," discusses information for specifying and manipulating user preferences.
- ❑ Chapter 5, "Window Managers," presents twm and mwm.
- ❑ Chapter 6, "Configuring the X Display Manager," contains specifics for setting up xdm.
- ❑ Chapter 7, "Font Maintenance," discusses adding fonts, aliasing, and rebuilding the font database.
- ❑ Appendix A describes common error messages.

Notational Conventions

The following conventions are used in this document:

- Monospace type indicates user-entered information for a computer program. You should enter these command sequences exactly as they appear.
- Italic type indicates user-specified variables.
- Brackets ([ ]) designate optional entries. Brackets are also used to distinguish parts of command strings that may be omitted.
- A horizontal ellipsis (. . .) shows repetition of the preceding item(s).
- A vertical ellipsis shows continuation of a sequence where not all of the statements in an example are shown.
- Clients and utilities that are documented in both the CONVEX OSF/Motif and CXwindows Programmer's References that include a number enclosed in parentheses refer to the appropriate section of the manual (for example, xload(1) means that documentation for the xload client is in Section 1 of the CXwindows Programmer's Reference).
- The | symbol is used in command sequences where you must pick no more than one alternative from a list of command options. For example, in the following command sequence,

  (fp)> s [et] s [pu-selftest] = [d [isable] | e [nable]]

  you must choose either d [isable] or e [nable].

- The percent symbol (%) signifies the standard C shell user prompt.

Note

A Note has information that may be of particular interest regarding the software or your files.

Associated Documents

Using this software successfully may require information not specific to the tasks described herein or not within the scope of this document.

CONVEX Computer Corporation provides the following O'Reilly & Associates, Inc. manuals and man pages to help you with CXwindows:

- *Volume 1, Xlib Programming Manual.*
- *Volume 2, Xlib Reference Manual.*
- *Volume 3, X Window System User's Guide.*
- *Volume 4, X Toolkit Intrinsics Programming Manual.*
- *Volume 5, X Toolkit Intrinsics Reference Manual.*
- CXwindows Programmer's Reference.

CONVEX Computer Corporation provides the following manuals and man pages to help you with CONVEX OSF/Motif:

- *OSF/Motif Programmer's Guide.*
- *OSF/Motif Style Guide.*
- *The X Window System: Programming and Applications with Xt, OSF/Motif Edition.*
- CONVEX OSF/Motif Programmer's Reference.

Ordering Documentation

To order the current edition of this or any other CONVEX document, send requests to:

CONVEX Computer Corporation
Customer Service
P.O. Box 833851
Richardson, TX 75083-3851 USA

Include the order number or the exact title.

In some situations, the current edition may not be desired. To receive a specific edition of a manual, contact the local CONVEX office or call the Technical Assistance Center.

Technical Assistance

If you have questions that are not answered in this book, contact the CONVEX Technical Assistance Center (TAC). To contact the TAC, use one of the following phone numbers:

- ❑ Within the continental U.S. - 1 (800) 952-0379
- ❑ Outside continental U.S. - Contact local CONVEX office

Reader Comments

If you have comments or questions about the contents of this book, please notify the CONVEX documentation department by using the Reader Comments form at the front of this document.

Product Overviews 1

CXwindows V2.0

CXwindows V2.0 includes all core clients from X11 Release 4 except xinit and xmh. In addition, it includes some clients developed by CONVEX, a number of contributed clients, and some utility programs. CXwindows also includes the standard X programming libraries as well as man pages for each supported client, utility, and library call.

Supported Core Clients

CXwindows V2.0 supports the following clients:

- appres - a utility for reviewing the application resource database.
- atobm - a conversion utility for bitmaps.
- bitmap - a bitmap editor.
- bmtoa - a conversion utility for bitmaps.
- listres - a utility for listing resources in widgets.
- oclock - a round clock.
- twm - a window manager.
- xauth - an X authority file utility.
- xbiff - a mailbox utility.
- xcalc - a scientific calculator.
- xclipboard - a utility that collects and displays text.
- xclock - a digital clock.
- xditview - a utility that displays ditroff files.
- xdm - the X display manager.
- xdpr - a utility for printing X window dump files.
- xdpyinfo - a utility to display information about X servers.
- xedit - a text editor.
- xfontsel - a point and click interface for selecting font names.
- xhost - a server access control program.

- xkill - a utility that kills a client by its X resource.
- xload - the load average display.
- xlogo - the logo for X.
- xlsatoms - a utility that lists interned atoms defined in server.
- xlsclients - a utility that lists client applications running on a display.
- xlsfonts - a utility to list available fonts.
- xlswins - a server window list display.
- xmag - a utility that magnifies parts of the screen.
- xman - a manual page display.
- xmodmap - a utility that modifies keymaps.
- xpr - a utility to print X window dums.
- xprop - a property display utility.
- xrdb - a resource database utility for the X server.
- xrefresh - a utility that refreshes all or part of an X screen.
- xset - a user preference utility.
- xsetroot - a set root window parameter utility.
- xstdcmap - a standard color map utility.
- xterm - a terminal emulator.
- xwd - a utility to do window dumps.
- xwininfo - a utility that provides window information.
- xwud - an image displayer.

Unsupported Clients

The latest standard release of X contains the following client applications. However, they are not included with CXwindows V2.0 nor are they supported by CONVEX.

- xinit - X server initialization.
- xmh - X interface to the MH mail system.

CONVEX Clients

The following client applications have been added to CXwindows and are fully supported:

- execqt - executes commands in background.
- xpostage - mailbox monitor.

Contributed Clients

The following public-domain clients are also included in CXwindows:

- ico - animates an icosahedron or other polyhedron.
- maze - automated maze.
- muncher - draws patterns in a window.
- plaid - paints plaid patterns in a window.
- puzzle - 15-piece puzzle game.
- xev - prints contents of X events.
- xeyes - follows the movements of the pointer.
- xgc - X graphics demo.

X Programming Libraries

CXwindows includes the following libraries in /usr/lib:

- libX11.a - contains Xlib C language programming library.
- libXaw.a - contains Athena widget set library.
- liboldX.a - contains X11 implementations of the X10 Xdraw and AssocTable routines.
- libXmu.a - contains routines that support the M.I.T. applications. The Athena widget set and some X clients use this library. The routines are not part of the Xlib standard.
- libXt.a - contains Xt Intrinsics library.
- libXext.a - contains routines for the following extensions: Multi-Buffering, SHAPE, and X Testing Consortium.
- libXinput.a - contains routines for XInputExtension.
- libXdmcp.a - contains routines for the xdm Control Protocol.
- libXau.a - contains routines for a sample authorization scheme.
- libX.a - contains a symbolic link to libX11.a for convenience.

Examples and Demos

The source to the demo programs can be found in /usr/lib/X11/examples/demos. As in previous releases of CXwindows, the M.I.T. Athena widget examples are in /usr/lib/X11/examples/Xaw and O'Reilly & Associates, Inc. Volume 1 examples are in /usr/lib/X11/examples/OReilly.

CONVEX OSF/ Motif V1.0

CONVEX OSF/Motif, CONVEX's implementation of the Open Software Foundation's user interface, is a separate product from CXwindows. But because CONVEX OSF/ Motif is a user environment based on the X Window System, CXwindows V2.0 is a prerequisite on your system. The CONVEX OSF/Motif environment consists of a toolkit, window manager, and user interface language (uil).

CONVEX OSF/Motif includes two OSF/Motif core clients, OSF/Motif programming libraries, library calls, include files, and man pages for each supported client and utility.

Supported Core Clients

CONVEX OSF/Motif V1.0 supports the following core clients:

- mwm - Motif Window Manager.
- uil - User Interface Language.

Refer to Chapter 5, "Window Managers," for information on mwm and its default configuration. Refer to the *OSF/Motif Programmer's Guide* and *The X Window System: Programming and Applications with Xt OSF/Motif Edition* for information on uil.

OSF/Motif Programming Libraries

CONVEX OSF/Motif includes the following libraries in /usr/lib:

- libXm.a - contains OSF/Motif widgets library.
- libXtM.a - contains OSF/Motif version of the Xt Intrinsics library.
- libUil.a - contains User Interface Language library.
- libMrm.a - contains OSF/Motif Resource Manager library.

The X Window System 2

Background

The X Window System, usually referred to as X, is a window environment designed for networked computer systems. It was developed at the Massachusetts Institute of Technology (M.I.T.) and first released in 1984.

X is run on a bitmap display (for example, a workstation or X terminal) and can show multiple applications at the same time. Each application appears in a separate window and might be running on a local host or a remote machine over the network. Windows can be moved around and stacked. Overlapping does not interfere with any activity occurring in the windows.

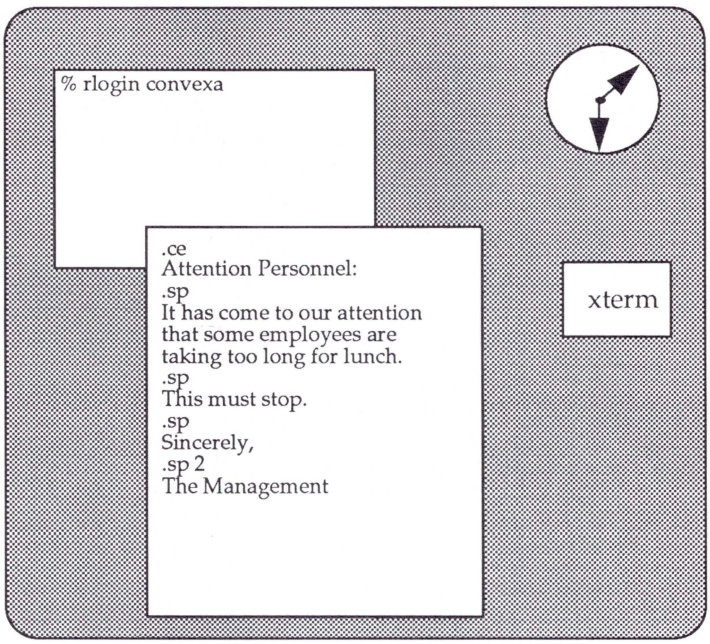
What can you do in these windows? Depending on the program running them, you might use windows as terminals to enter commands or edit a program. Others may function as simple output-only displays that show the time and date. Figure 2-1 shows an X display.

The X Display

The most frequently used window will probably be your xterm window. xterm windows are terminal emulators in which you perform actions you would normally do on a standard terminal. Figure 2-1 shows two xterm windows, a clock, and an icon.

Icons are small symbols that represent an inactive window. You can convert windows to icons and back to windows again. Converting windows to icons gives you more space on your display. Every icon has a label that is the name of the program that created the window. The contents of the window are not visible when it is iconified. And icons can be moved just like windows.

Figure 2-1
X Display



The shaded area behind the windows is your root, or background, window.

An important aspect of X is the window manager. Window managers control the general operation of the window system, allowing you to alter the size and position of windows.

Interactions

How do you get input into these windows? All X displays require you to use a pointer of some type, usually a mouse with several buttons. As you move the mouse on its pad, the cursor in your display mimics its movements. Depending on where the cursor is in the display, it can take on a different appearance. For example, when the cursor is in an xterm window, it looks like an "I," but when it is in the root window, it appears as an "X." Refer to Appendix D in the *X Window System User's Guide* for a complete list of cursors.

You can enter input in a window once you have focused on, or selected, it. The cursor must be within a window's borders and those borders highlighted before you type anything.

You also use the pointer to display menus. By holding down a specific button, menu items are displayed. As you slide the cursor up and down the menu, items are highlighted. When you have highlighted the item you want, release the button to select it.

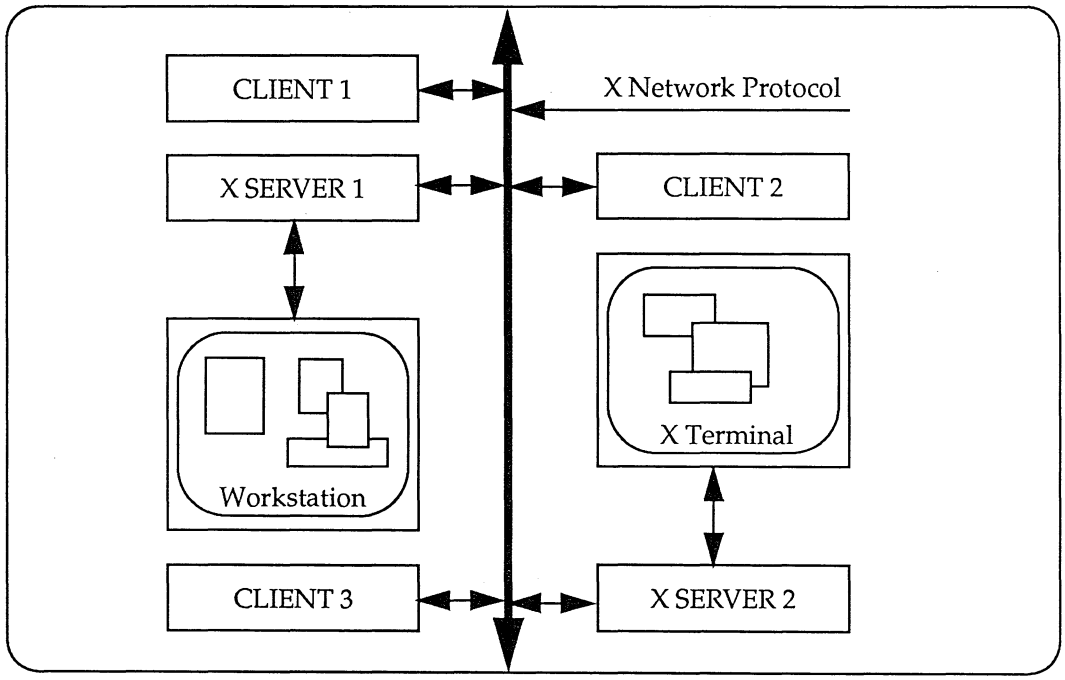
The Server-Client Model

Most window systems are kernel based and therefore closely tied to an operating system. This means that applications are operating-system dependent and can only run on a discrete system. X is not part of any operating system but comprised of user-level programs.

X is based on the server-client model. The server provides display capabilities and keeps track of all input. Clients are application programs that perform specific tasks. You could view it as the server being an intermediary between clients and the display hardware; clients make requests that the server communicates to the hardware display.

The division between client and server in X allows clients and the server either to work on the same machine or to reside on different machines connected by a network. For example, you might use a workstation as a server to interact with clients that are running on a CONVEX supercomputer. Even though the client is actually running on the supercomputer, all user input and output occur on the workstation server and are communicated across the network using the X protocol. Figure 2-2 shows a simple server-client network.

Figure 2-2
Server-client Example



Another advantage to the server-client model is that because the server is the only component responsible for interacting with the hardware, only the server program must be machine specific. Client programs can be easily ported from system to system.

Clients

X allows you to run many clients simultaneously on different computers on the network. It is important to know that identical programs may not look the same on different servers for the following reasons:

1. There is no standard user interface.
2. Users can customize clients differently on each server.
3. Hardware may be different on each server.

Several of the more frequently used clients are discussed in the following sections.

The Window Manager

A special client called a window manager is responsible for controlling use of space on the screen. The window manager is a user-interface mechanism and dictates a screen layout policy and implement a certain "look and feel." They let you move and resize windows and create additional windows among other things. Currently, CONVEX offers two window managers: twm with CXwindows and mwm with CONVEX OSF/Motif. Refer to Chapter 5, "Window Managers," and the twm(1) and mwm(1) man pages for more information.

The Terminal Emulator

You will run X on either a workstation or an X terminal. Both have large, bitmapped screens that allow for many windows. The most frequently used window will be your xterm window.

xterm windows are terminal emulators in which you perform actions you would normally do on a standard terminal (for example, enter commands, edit a file, or compile programs). For more information about xterm windows, refer to Chapters 2 and 4 of the *X Window System User's Guide*.

The Display Manager

The X display manager (xdm) manages a collection of remote displays from a single system. The xdm login window is the first interface users see. Refer to Chapter 6, "Configuring the X Display Manager," for more information on xdm and how to start it.

Other Clients

A complete list of clients for CXwindows and CONVEX OSF/Motif can be found in Chapter 1, "Product Overviews." All clients are located in /usr/bin/X11.

X Command Line Options 3

Common Options

X applications share a common set of command line options. Most of these options control the appearance of the application. X clients also have individual options that control the features of the application. For example, the xclock client has an option to run in analog or digital mode. This chapter presents an overview of the standard client options.

Specifying Fonts

Users can specify different fonts when displaying the windows, menus, etc. of many clients. X supports many different display fonts, sizes, and type styles.

Note

This discussion deals with screen fonts and should not be confused with printer fonts.

With most clients, you can specify the display font as a command line option. The standard X Toolkit option to specify fonts for text display is `-fn`. For example, the following command line creates an xterm window where text is displayed with a font name you supply:

```
% xterm -fn font_name &
```

Some clients require that you specify the font in a resource or initialization file. This is the case with twm menus where fonts are specified in your .twmrc (resource file in your home directory for twm). Refer to the relevant man page on each client for information on specifying display fonts.

For the list of standard R4 fonts, type `xlsfonts` in an xterm window. You can also use the xfontsel(1) program to browse through the available font styles.

Specifying Other Options

There are numerous standard options that all X applications recognize. The following list briefly describes each one. Always refer to an application's man page for a complete list of available options.

- `-background` specifies the background color of the window. You can also use the `-bg` flag to specify background color.
- `-bd` specifies the border color of the application window. The option `-bordercolor` may also be used.
- `-bw` specifies the border width of the application window.
- `-display` indicates the display that the client should run on. This option is often abbreviated as `-disp`.
- `-font` specifies the font to be used for text display. It is usually abbreviated as `-fn`.
- `-foreground` specifies the foreground (drawing or text) color of the application. This option can be abbreviated as `-fg`.
- `-geometry` specifies the preferred size and position of the application window.
- `-help` indicates that a brief summary of the allowed options should be printed on the standard error.
- `-iconic` specifies that the application should start in iconic form. An application will not start in iconic form unless a window manager is already running.
- `-name` allows you to change the name of the application. This can be very useful: different invocations of the same application can use different values for the X defaults. Refer to Chapter 4, "X Application Defaults," for more information.
- `-reverse` specifies that the application should run in reverse-video mode (foreground and background colors are swapped). The option may be abbreviated as `-rv`. The option `+rv` specifies that reverse-video should not be used.
- `-title` assigns a title to the window. Depending on the window manager and your window manager configuration, the title may not be shown. By default, twm and mwm display titles.
- `-xrm` specifies a resource to be loaded into the resource manager. The argument to this option is a quoted string similar to the lines found in a resource file. Refer to Chapter 4, "X Application Defaults," for more information.

X Application Defaults 4

Resources and Preferences

X applications are highly configurable. You can specify the fonts and colors of an application, control the size and placement of an application's windows, and change the bindings of your keyboard. By specifying your own preferences for these parameters, you can customize the behavior of applications.

Features that you customize are called user preferences and are specified using resources. Your preferences are stored in a database in the file $HOME/.Xdefaults. When you start X, your resource database is loaded into the X server where all applications can access the information.

In addition to personal preferences, there are system-wide resources for each application; these resources are used as your preferences if you do not specify any. The directory named /usr/lib/X11/app-defaults contains all system-wide resource files.

Specifying Resources

Resources, also called X defaults, are simply *<name, value>* pairs that control the appearance or function of a particular program. Resources provide a convenient way to tailor whole collections of interface objects and applications with a minimal amount of work.

Resources are specified as

program.name.subname.etc: value

one per line in resource files. Resource names are hierarchies, corresponding to major structures within an application (where structures are often objects like windows, panels,

menus, etc.). Resource names can also be application-specific tunable parameters.

The various subnames of resources are called components and are specified left to right from most general to least general.

Instances and Classes

The visible results of a resource specification depend on whether the specification is for an individual resource or for a class of resources.

For example, *foreground* refers to the foreground color of text and lines; *foreground* is a specific resource. *Foreground* refers to the entire class of foreground resources (such as window border color, pointer color, and cursor color); *foreground* is an instance of the class *Foreground.*

Note

Instance name components begin with lowercase letters and class name components begin with uppercase letters.

The distinction between classes and instances is important. For example, the specification **borderColor: blue* would make all window borders blue. However, the line **Foreground: blue* would make all resources that belong to the *Foreground* class blue, including border color, pointer color, etc. The "*" is a wildcard that implies "any."

Classes can be viewed as sets, and each instance is an element of that set. The class *Foreground* includes window border color as a member.

Class and instance resources can be combined in powerful ways. For instance, many text applications have some notion of background, foreground, border, pointer, and cursor or marker color. Usually the background is set to one color, and all of the other attributes are set to another so that they may be seen on a monochrome display. To allow users of color displays to set any or all of them, the colors may be organized into classes as follows:

Instance Name	Class Name
background	Background
foreground	Foreground
borderColor	Foreground
pointerColor	Foreground
cursorColor	Foreground

To configure the application for monochrome mode with two colors, you would use two resource specifications:

```
*Foreground: red
*Background: blue
```

Then, if you want the cursor to be yellow, but the pointer and the border to remain the same as the foreground, you would need only one new resource specification:

```
*cursorColor: yellow
```

Because class and instance names are distinguishable by case, both types of names may be given in a single resource specification. Here are some additional examples:

```
xedit*background: red
*command.font: 8x13
*Font: 8x13
xedit*Command*activeForeground:black
```

Components that consist of more than one word (such as *activeForeground)* have the succeeding words capitalized and concatenated without any spaces. An instance of an icon pixmap might be called *iconPixmap* whereas the class of icon pixmaps would be called *IconPixmap*. The capitalization is important because resource names may contain both instance and class name components within the same specification (as in *xedit*paned*Command.activeForeground: black*).

Specifying Resources for the xedit Client

The internal structure of interface objects within an application dictates the order of components in a resource name. Components may be class names or instance names but must be specified in an order that the application will recognize.

In this section, we will use xedit as an example and show how to specify its resources.

The xedit application contains a hierarchy of the following user interface objects:

Xedit xedit
    Paned paned
        Paned buttons
            Command quit
            Command save
            Command load
            Text file_name
        Label bc_label
        Text messageWindow
        Label labelWindow
        Text editWindow

At the top of the hierarchy is a large paned window. It in turn contains another paned window, two text windows, and two labels. The paned window called *buttons* contains command buttons. The *save* button (used to save files) could be named as follows:

Program Name	Pane Name	Object Group Name	Subobject Name
xedit	*paned*	*buttons*	*save*

xedit.paned.buttons.save is a fully-specified name. If we wanted to change the font used to display the button label to 8x13, we would specify the resource

```
xedit.paned.buttons.save.font: 8x13
```

In the above example, if we assume that xedit is one of several X editor types of programs, we could build the following class name, where *Xedit* is the name of all X editor programs:

Application Type	Top Level Area Type	Second Level Object Type	Third Level Object Type
Xedit	*Paned*	*Paned*	*Command*

To change the font used in all the buttons to 8x13, we could specify the resource

```
Xedit.Paned.Paned.Command.font: 8x13
```

Class names allow default values to be specified for all versions of a given object. Instance names allow a value for a particular version to be given that overrides the class value and can be used to specify exceptions to the rules outlined by the class names. For example, we could specify that all command buttons in paned windows should have a background color of blue except for button *save* (it should be red):

```
*Paned.Paned.Command.Background: blue
xedit.paned.buttons.save.background: red
```

Resource name hierarchies do not have to be fully specified. In the preceding example, each of the individual components was listed. This can be cumbersome. Instead of giving a full specification for each set of objects (there might be sliders, edit windows, or any number of other types), the "*" separator can be used as a wildcard to omit intervening components. Rewriting the previous example yields:

```
xedit*Paned*Background: blue
xedit*paned.buttons.save.background: red
```

In general, it is a good idea to use the "*" instead of the "." in case you have forgotten any intervening components or in case new levels are inserted into the middle of the hierarchy.

Specific Resources for xedit

In addition to the resources associated with the structure of xedit, this application also has three user-configurable parameters that control how it works:

- *enableBackups*. This resource specifies that when changes are made to an existing file, xedit is to copy the original version to <prefix>*filename*<suffix> before it saves the changes. The default for this resource is "off;" no backups are to be done
- *backupNamePrefix*. This resource specifies a string to be prepended to the backup file name. By default, the string is empty (no string is prepended).
- *backupNameSuffix*. This resource is similar to *backupNamePrefix* but specifies a string to be appended to the backup file name. By default, the string is ".BAK."

Writing Resources for Other X Clients

The internal hierarchy of each application is different. Refer to each client's man page for a description of the program's structure and other application-specific configurable resources.

Precedence Rules for Resources

When using X resources, a more specific resource always takes precedence over a more general one. For example, assume you have the following lines in your .Xdefaults file:

Figure 4-1
.Xdefaults File

```
*Foreground: white
*Background: black
xterm*Foreground: black
xterm*Background: white
xterm*pointerColor: yellow
xclock.hands: red
```

- The first line makes all resources of class *Foreground* white.
- The second line makes all resources of class *Background* black. These two lines are used to make applications appear in reverse video (white on black).
- The third and fourth lines overrule the first and second lines but only in the case of xterm. For xterm windows, all foreground objects (except *pointerColor*) will be displayed in black while all background objects will be displayed in white.
- On the fifth line, the color of the pointer in an xterm window will be yellow. This specification overrides line three.
- Finally, the specification on the last line overrides the general specification on line one. The hands of an xclock are of class *Foreground*, but because *hands* is a specific instance resource, the hands will be red.

Using xrdb to Configure the X Server

In X11 Release 4, the window property mechanism in the X protocol stores resources in the server where they are available to all clients. xrdb is used to get or set the contents of the RESOURCE_MANAGER property on the root window of screen 0. You normally run xrdb from your X startup file, .xsession. With the resource information in the server, resources may be dynamically specified based on the particular display being used. This is useful in setting up different defaults for monochrome and color displays.

xrdb uses the C preprocessor to provide conditionals based on the capabilities of the server you are using. Refer to Chapter 7 in the *X Window System User's Guide* and the xrdb(1) man page for more information.

Setting Resources

In X11 you have a lot of flexibility concerning the location of resource definitions. The Resource Manager obtains resource specifications from:

- Application-specific resource files stored in /usr/lib/X11/app-defaults.
- Application-specific resource files in the directory named by the environment variable XAPPLRESDIR (default value is $HOME) for programs written with the X Toolkit.
- RESOURCE_MANAGER property on the root window of screen 0 (stored by using xrdb). If this property is not defined, then $HOME/.Xdefaults is read.
- User-specific defaults stored in a file whose name is set in the environment variable XENVIRONMENT.
- Command line option (`-xrm`) for programs written with the X Toolkit.

Resources are usually loaded from a file into the RESOURCE_MANAGER property with xrdb from the script you use to start X. Sites that use xdm should load user-specified resources automatically. Refer to Chapter 6, "Configuring the X Display Manager," for more information on xdm.

Example Scenario

User preferences are stored in the file $HOME/.Xdefaults. Assuming that you start X using xdm, the example files in Figures 4-2 and 4-3 would load the X server with your resource database and start a set of X applications.

Figure 4-2
.xsession File

```
#!/bin/sh
xrdb -load $HOME/.Xresources
xclock -geometry 48x48-1+1 &
xload -geometry 48x48-1+100 &
xterm -geometry 80x55+0+0 &
xterm -geometry 80x65+488+1 &
xterm -geometry 80x20+0-0 &
xload =48x48-1+150 &
xbiff -rv =48x48-1+50 &
exec twm
```

Figure 4-3
.Xdefaults File

```
XTerm*Foreground: white
XTerm*Background: #c00
XTerm*BorderColor: white
tinyxterm*Font: 3x5
tinyxterm*Geometry: 80x24
bigxterm*Font: 9x15
bigxterm*Geometry: 80x55
xload*Background: black
xload*Foreground: red
xbiff*borderWidth: 0
xbiff*Background: white
xbiff*Foreground: blue
xbiff*reverseVideo: on
```

Note the alternating instance names `tinyxterm` and `bigxterm`. Different resources can be specified for different instances of an X client. Instance names for X clients are specified with the `-n` option. The command

```
xterm -name tinyxterm
```

invokes an 80-character column by 24-character row xterm window using the 3x5 font, while

```
xterm -name bigxterm
```

invokes an 80-character column by 55-character row xterm window and uses the 9x15 font.

Window Managers 5

The Current View

With Release 4 of X11, M.I.T. no longer supports awm or uwm. Following this decision, CONVEX no longer supports either of these window managers. CONVEX recommends you change your window manager to either twm or mwm. Both are fully-compliant window managers and can be used with advanced window-management tools like room and session managers. Both are fully supported by CONVEX and explained in this chapter.

twm

twm is a highly-flexible window manager packaged with X and offers powerful window-management features like:

- Title bars that display the name of the application running in a window.
- Windows that are highlighted when you select them.
- Icons and an icon manager to keep track of them.
- Move, resize, and freeze-focus features in the title bar.

When invoked, twm looks in your home directory for a start up file called .twmrc. The .twmrc file contains sections that define variables, buttons, and menus. These sections can appear in any order.

The following sections show the default .twmrc CONVEX provides located in /usr/lib/X11/system.twmrc. Due to the length of the .twmrc, it is broken across several pages for documentation purposes only. For detailed information, refer to the twm(1) man page.

```
# Icon Manager Variables
DontIconifyByUnmapping { "TWM Icon Manager" }
ForceIcons
IconifyByUnmapping
IconManagerDontShow { "xbiff" "xclock" "oclock" "xload" "xpostage" }
IconManagerGeometry "400x200+160-10" 3
NoCaseSensitive
ShowIconManager
SortIconManager
Zoom 8
#
# Window Behavior Variables
AutoRaise { "TWM Icon Manager" }
AutoRelativeResize
ClientBorderWidth
DontMoveOff
NoGrabServer
DecorateTransients
NoTitle { "xbiff" "xeyes" "xload" "oclock" "xpostage" "xclock" "TWM Icon Manager"
}
RestartPreviousState
UsePPosition "non-zero"
#
# Font Variables
TitleFont       "-adobe-helvetica-bold-r-normal--*-120-*-*-*-*-*-*"
ResizeFont      "-adobe-helvetica-bold-r-normal--*-120-*-*-*-*-*-*"
MenuFont        "-adobe-helvetica-bold-r-normal--*-120-*-*-*-*-*-*"
IconFont        "-adobe-helvetica-bold-r-normal--*-100-*-*-*-*-*-*"
IconManagerFont "-adobe-helvetica-bold-r-normal--*-100-*-*-*"
#
# Menu Variables
DefaultFunction f.beep
# Color Variables (Color used only if display has color;
#       Monochrome used for monochrome displays)
Color
{
 BorderColor            "slategrey"
 DefaultBackground      "maroon"
 DefaultForeground      "gray85"
 TitleBackground        "maroon"
 TitleForeground        "gray85"
 MenuBackground         "maroon"
 MenuForeground         "gray85"
 MenuTitleBackground    "gray70"
 MenuTitleForeground    "maroon"
 IconBackground         "maroon"
 IconForeground         "gray85"
 IconBorderColor        "gray85"
 IconManagerBackground  "maroon"
 IconManagerForeground  "gray85"
}
InterpolateMenuColors
Monochrome { BorderColor "black" }
```

Buttons

```
#
# Root window button functions
#
Button1 =       : r             : f.menu "General"
Button2 =       : r             : f.menu "Local xterms"
Button3 =       : r             : f.menu "X clients"
#
# Title, Frame, Icon button functions
#
Button1 =       : i|t           : f.menu "Window Ops"
Button2 =       : i|t|f         : f.move
Button3 =       : i|t|f         : f.raiselower
#
# Iconmgr button functions
#
Button2 =       : m             : fmenu "Window Ops"
Button3 =       : m             : f.raiselower
#
# All windows menu
#
Button1 =       : m    : all    : f.menu "Window Ops"
Button2 =       : m    : all    : f.move
Button3 =       : m    : all    : f.iconify
```

```
# Menu Definitions
#
Menu "Window Ops"
{
 "iconify"      f.iconify
 "raise"        f.raise
 "lower"        f.lower
 "move"         f.move
 "destroy"      f.destroy
 "autoraise"    f.autoraise
 "identify"     f.identify
 "refresh"      f.winrefresh
 "zoom"         f.menu "zoom functions"
}
Menu "General"
{
 "manager"                f.title
 "iconify"                f.iconify
 "destroy"                f.destroy
 "focus"                  f.focus
 "identify"               f.identify
 "lower"                  f.lower
 "raise"                  f.raise
 "move                    f.move
 "refresh window"         f.winrefresh
 "refresh display"        f.refresh
 "resize"                 f.resize
 "more twm functions"     f.menu "twm functions"
 "TWM Windows"            f.menu "TwmWindows"
 " "                      f.nop
 "restart twm"            f.restart
 "quit twm"               f.quit
}

Menu "X clients"
{
 "X clients"    f.title
 "oclock"       !"oclock &"
 "xbiff"        !"xbiff &"
 "xcalc"        !"xcalc &"
 "xclock"       !"xclock &"
 "xeyes"        !"xeyes &"
 "xmag"         !"xmag &"
 "xpostage"     !"xpostage &"
}
```

```
Menu "twm functions"
{
 "twm functions"        f.title
 "f.zoom functions"     f.menu "zoom functions"
 "autoraise"            f.autoraise
 "f.beep"               f.beep
 "f.circledown"         f.circledown
 "f.circleup"           f.circleup
 "f.deiconify"          f.deiconify
 "f.forcemove"          f.forcemove
 "f.nop"                f.nop
 "f.raiselower"         f.raiselower
 "f.unfocus"            f.unfocus
}
Menu "zoom functions"
{
 "zoom functions"       f.title
 "fill vertically"      f.zoom
 "fill horizontally"    f.hzoom
 "fill top half"        f.htzoom
 "fill bottom half"     f.hbzoom
 "fill left side"       f.vlzoom
 "fill right side"      f.vrzoom
 "fill full screen"     f.fullzoom
}
Menu "Local xterms"
{
 "xterms"               f.title
 "xterm"                !"xterm -n `hostname` &"
 "reverse video xterm"  !"xterm -n `hostname` -rv &"
 "xterm font 8x13"      !"xterm -font 8x13 -n `hostname` &"
 " "                    f.nop
 "remote xterms"        f.menu "Remote xterms"
}
Menu "Remote xterms"
{
 "remote xterms"   f.title
 "convexa"         !"rsh convexa -n execqt xterm -disp $DISPLAY -n convexa &"
 "convexb"         !"rsh convexb -n execqt xterm -disp $DISPLAY -n convexb &"
 "convexc"         !"rsh convexc -n execqt xterm -disp $DISPLAY -n convexc &"
}
```

mwm

mwm is packaged with CONVEX OSF/Motif and provides you with a very popular user interface, a high degree of flexibility, and a pleasing visual interface.The Resource Manager loads the following application defaults from /usr/lib/X11/app-defaults/Mwm before looking in your home directory for your default files.

```
!
! app-defaults RESOURCE SPECIFICATIONS FOR Mwm
!

*moveThreshold:              3
Mwm*buttonBindings:          DefaultButtonBindings
Mwm*keyBindings: DefaultKeyBindings

!
! component appearance resources
!
*resizeBorderWidth:          7

! FONT stuff
*fontList:                   variable
! Use smaller fixed font for icons
*icon*fontList:              fixed

!
! END OF RESOURCE SPECIFICATIONS
!
```

The following pages show the default .mwmrc CONVEX provides located in /usr/lib/X11/system.mwmrc. Similar to .twmrc, it describes buttons, keys, and menus. For detailed information, refer to the mwm(1) man page and sections I.3.8 and II.4 of the *OSF/Motif Programmer's Guide.* Due to the length of the .mwmrc, it is broken into two sections (the following two pages) for documentation purposes only.

```
# DEFAULT mwm RESOURCE DESCRIPTION FILE (system.mwmrc)
#
# Description:
#       This is the default mwm resource description file. mwm will use
#       this file if it cannot find a ".mwmrc" file in your home directory.
#       To change the default configuration of your mwm, copy this file
#       into ".mwmrc" in your home directory, edit it, and restart mwm.
#
# menu pane descriptions
#
# Root Menu Description (button 1)
#
Menu RootMenu
{
 "Root Menu"      f.title
 "New Window"     f.exec "xterm &"
 "Circle Up"      f.circle_up
 "Circle Down"    f.circle_down
 "Refresh"        f.refresh
 no-label         f.separator
 "Restart..."     f.restart
 "Exit"           f.quit_mwm
}
# rlogin Menu Description (button 2)
#
Menu RloginMenu
{
 "Remote Logins"  f.title
 "machine1"       !"rsh machine1 execqt xterm -d $DISPLAY -ls -n machine1"
 "machine2"       !"rsh machine2 execqt xterm -d $DISPLAY -ls -n machine2"
 no-label         f.separator
 "First Group"    f.menu RloginMenu1
 "Second Group"   f.menu RloginMenu2
}
# rlogin Menu 1
#
Menu RloginMenu1
{
 "First Group"    f.title
 "mach1a"         !"rsh mach1a execqt xterm -d $DISPLAY -ls -n mach1a"
 "mach1b"         !"rsh mach1b execqt xterm -d $DISPLAY -ls -n mach1b"
 "mach1c"         !"rsh mach1c execqt xterm -d $DISPLAY -ls -n mach1c"
}
# rlogin Menu 2
#
Menu RloginMenu2
{
 "Second Group"   f.title
 "mach2a"         !"rsh mach2a execqt xterm -d $DISPLAY -ls -n mach2a"
 "mach2b"         !"rsh mach2b execqt xterm -d $DISPLAY -ls -n mach2b"
 "mach2c"         !"rsh mach2c execqt xterm -d $DISPLAY -ls -n mach2c"
}
```

```
# Utility Menu (button 3)
Menu UtilityMenu
{
 "Utility Menu"   f.title
 "xterm"          !"xterm -display $DISPLAY"
 "xclock"         !"xclock -display $DISPLAY"
 "xbiff"          !"xbiff -display $DISPLAY"
 "xcalc"          !"xcalc -display $DISPLAY"
 "xload"          !"xload -display $DISPLAY"
 "xlogo"          !"xlogo -display $DISPLAY"
}
# Default Window Menu Description
Menu DefaultWindowMenu MwmWindowMenu
{
 "Restore"   _R   Alt<Key>F5     f.normalize
 "Move"      _M   Alt<Key>F7     f.move
 "Size"      _S   Alt<Key>F8     f.resize
 "Minimize"  _n   Alt<Key>F1     f.minimize
 "Maximize"  _x   Alt<Key>F2     f.maximize
 "Lower"     _L   Alt<Key>F3     f.lower
 no-label                        f.separator
 "Close"     _C   Alt<Key>F4     f.kill
}
# key binding descriptions
#

Keys DefaultKeyBindings
{
 Shift<Key>Escape              icon|window          f.post_wmenu
 Meta<Key>space                icon|window          f.post_wmenu
 Meta<Key>Tab                  root|icon|window     f.next_key
 Meta Shift<Key>Tab            root|icon|window     f.prev_key
 Meta<Key>Escape               root|icon|window     f.next_key
 Meta Shift<Key>Escape         root|icon|window     f.prev_key
 Meta Ctrl Shift<Key>exclam    root|icon|window     f.set_behavior
 Meta<Key>F6                   window               f.next_key transient
}
#
# button binding descriptions
#
Buttons DefaultButtonBindings
{
 <Btn1Down>         frame|icon     f.raise
 <Btn2Down>         frame|icon     f.post_wmenu
 <Btn1Down>         root           f.menu RootMenu
 <Btn2Down>         root           f.menu RloginMenu
 <Btn3Down>         root           f.menu UtilityMenu
 Meta<Btn1Down>     icon|window    f.raise
 Meta<Btn2Down>     icon|window    f.lower
 Meta<Btn3Down>     icon|window    f.move
}
# END OF mwm RESOURCE DESCRIPTION FILE
```

Configuring the X Display Manager

6

Control Protocol

The X display manager, xdm, provided in CXwindows V2.0 supports the X Display Manager Control Protocol (XDMCP) to control remote displays. In order to use XDMCP, your X server must also support it. Before proceeding, please familiarize yourself with the features of your X display server and determine if it supports XDMCP. Consult your X server's man page or the configuration instructions for your X terminal. Once you understand the capabilities of your server, you can proceed with configuring xdm on the CONVEX.

The current version of xdm is backward-compatible with previous releases. If you are already using an earlier version of xdm, you are not required to make any changes to use the newest version of xdm. xdm can also support XDMCP and non-XDMCP servers simultaneously. Refer to the section on configuring xdm for non-XDMCP servers and the Xservers file for more information.

The xdm daemon can be found in /usr/bin/X11. xdm can only be run by the superuser and should be invoked at boot time. More information on how to invoke xdm is given at the end of this chapter.

The X Display Manager

xdm manages a collection of remote X displays from your CONVEX supercomputer. xdm provides services similar to those provided by init(8), getty(8) and login(1) on character terminals: xdm prompts for a login ID and password, authenticates the user, and starts the user's (or a default) X session.

The X session script invokes X clients such as a terminal emulator and a clock and then starts the user's window manager. When the user quits the window manager, all of the windows are destroyed and the session ends.

When an X session is terminated, xdm resets the X server on the remote display and restarts the entire process over again.

Because xdm provides the first interface that users will see, it is designed to be simple to use and easy to customize to meet the needs of your site. xdm has many options. Refer to the xdm(1) man page for a complete list of options and resources.

xdm Configuration and Setup Files

xdm has a number of configuration files. These files are located in /usr/lib/X11/xdm.

xdm-config

The actions of xdm can be controlled using the configuration file xdm-config; the contents of the file are in the familiar X resource format. Some resources modify the behavior of xdm on all displays, while others modify its behavior on a single display or class of displays.

Resource names that are specific to a single display have the display name inserted into the resource name between *DisplayManager* and the final resource name component. For example, *DisplayManager.fate_0.startup* is the name of the resource that defines the startup shell file on the *fate:0* display. Because the resource manager uses colons to separate the name of a resource and its value, and dots to separate resource name components, xdm substitutes underscores for the dots and colons when generating the resource name.

Figure 6-1 shows part of the contents of a sample xdm-config file:

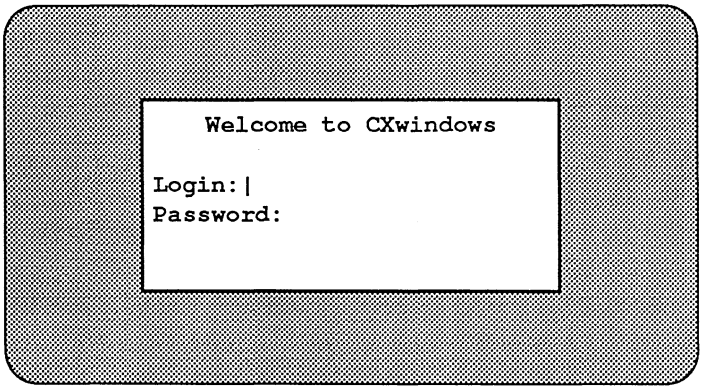
- The line *DisplayManager.errorLogFile* directs error output to the file /usr/lib/X11/xdm/xdm-errors. This file will also contain any output directed to standard error (stderr) by Xstartup, Xsession, and Xreset; examine the errorLogFile when debugging these scripts. If this resource is not set, error output is directed to the system console.
- *DisplayManager*resources* specifies the name of a resource file to be loaded by xrdb onto the root window of all dis-

Figure 6-1
xdm-config File

```
DisplayManager.errorLogFile:  /usr/lib/X11/xdm/xdm-errors
DisplayManager*resources:     /usr/lib/X11/xdm/Xresources
DisplayManager*startup:       /usr/lib/X11/xdm/Xstartup
DisplayManager*reset:         /usr/lib/X11/xdm/Xreset
DisplayManager*session:       /usr/lib/X11/xdm/Xsession
DisplayManager.fate_0.resources: /usr/lib/X11/xdm/fateXresources
DisplayManager.ncda_0.resources: /usr/lib/X11/xdm/ncdaXresources
```

plays unless otherwise specified (as *DisplayManager.fate_0.resources* does). This resource database is loaded just before the authentication procedure is started so it can control the appearance of the xdm window. An example xdm login window is shown in Figure 6-2.

Figure 6-2
xdm Login Window



The Xresources file in Figure 6-3 was used to create the login window above:

Figure 6-3
Xresources File

```
xlogin*Font:       -bitstream-charter-medium-r-normal--17-120-100-
100-p-95-iso8859-1
xlogin*borderWidth:5
xlogin*fail:       login failed, please try again...
xlogin*failFont:   -adobe-new century schoolbook-medium-r-normal--18*
xlogin*greetFont:  -adobe-new century schoolbook-bold-r-normal--18*
xlogin*greeting:   Welcome to CXwindows
```

- *DisplayManager*startup* specifies a program or script that is run as root after the authentication process succeeds. By default, no program is run. Refer to the section on Xstart-

up below.

- *DisplayManager*session* specifies a session to be executed by the user. By default, xterm is run. Refer to the Xsession section further on in this chapter.
- *DisplayManager*reset* specifies a program or script that is run as root after the session terminates. By default, no program is run. More information is given in the Xreset section.
- *DisplayManager.fate_0.resources* and *DisplayManager.ncda_0.resources* specify display-specific resources for the display *fate:0* and *ncda:0*, respectively. A display-specific resource file might change the login greeting or the fonts used. Figure 6-4 is an example of the fateXresources file.

Figure 6-4
fateXresources File

```
xlogin*borderWidth: 3
xlogin*fail: login failed, please try again...
xlogin*failFont: -adobe-new century schoolbook-medium-r-normal--18*
xlogin*font: -bitstream-charter-medium-r-normal--17*
xlogin*greetFont: -adobe-new century schoolbook-medium-r-normal--18*
xlogin*greeting: This is Fate's Workstation
```

Xstartup

This file is typically a shell script. This script is the place to mount user's home directories from file servers, display the message of the day, or abort the session if no logins are allowed. It is run as root so you should be very careful about security. An example Xstartup file is in Figure 6-5.

Figure 6-5
Xstartup File

```
#!/bin/sh
#
# Xstartup
# This program is run as root after the user is verified.
# If this script exits with a nonzero return code, the
# session is immediately terminated and xdm returns
# to the login window

if [ -f /etc/nologin ]; then
 exit 1
fi
exit 0
```

This script checks if /etc/nologin exists and, if it does, terminates the session. This prevents users from starting sessions when they are not permitted to log onto the system.

Various environment variables are set up for the use of this script:

- `DISPLAY` is set to this display's name. For example, if a user was logging in from the X terminal named *fate:0*, `DISPLAY` would be set to *fate:0*.
- `HOME` is set to the home directory of the user.
- `USER` is set to the name of the user.

No arguments are passed to the Xstartup script. xdm waits for the script to exit before starting the user's session. If the script exits with a non-zero status, xdm discontinues the session immediately and restarts the authentication process.

Xsession

This is the script that is run as the user's X session. It is run with the permissions of the user and has the same environment variables defined as the Xstartup script. However, `SHELL` is set to the user's login shell as defined in the file /etc/passwd.

The Xsession file should look in $HOME for a user's personal .xsession file. The user's file would specify a list of clients that the user would like to use as a session. The user's session file should replace the system default standard. If the user does not have a session file, the default session should be invoked and should provide a window manager and an xterm window.

Figure 6-6 shows an example Xsession file. If the user has a .xsession file, it is executed using the user's login shell. If there is no session file for the user, the user's .Xdefaults file is loaded into the server, and an xterm window and twm are started.

Figure 6-6
Xsession File

```
#!/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 user
# .xsession file instead of forcing a particular
# session layout
#

if [ -f $HOME/.xsession ]; then
 exec $SHELL $HOME/.xsession
else
 if [ -f $HOME/.Xdefaults ]; then
 xrdb -load $HOME/.Xdefaults
 fi
 execqt xterm -fn 10x20 -geometry 80x24+20+20 -ls
 exec twm
fi
```

When the user exits the window manager, the session is terminated.

Xreset

This script is run after the user session has terminated. The script is run as root and should contain commands that undo the effects of whatever commands the Xstartup script ran. The same environment variables that were passed to Xstartup are passed to Xreset.

Refer to the xdm(1) man page for more examples of script files and resource configurations.

Running xdm Servers with XDMCP

When an X server with XDMCP boots, it contacts its xdm host. In turn, the host responds by displaying the xdm login window on the display. Depending on the configuration, an XDMCP server can ask one or more machines for a login window.

Servers that have XDMCP initiate the xdm exchange. The servers request a login window from the xdm host and the xdm host answers the request. In particular, xdm:

1. Identifies the display initiating the request.
2. Examines the xdm-config file for any display-specific resources. If there are resources defined for that display, xdm loads those resources into the server. If there are no display-specific resource, xdm loads the default Xresources file into the server.
3. Displays the xdm login window and starts the authentication procedure.

If a user enters the correct login ID and password, xdm starts the X session on that display.

To support XDMCP servers you must:

- Invoke the X server or configure the X terminal to contact your xdm host machine at server boot time.
- Create any display-specific resources you want.

Adding New Servers

When adding new servers, you will probably add new display-specific resources to the xdm-config file. To have the changes become effective immediately, you must manually reconfigure the xdm daemon. To reconfigure xdm, run the command sequence shown in Figure 6-7.

Figure 6-7
Reconfiguring xdm

```
% ps aux | grep xdm
root 6777 0.3 0.1 172 52 q5 S 0:00 grep xdm
root 198 0.0 0.0 692 4 ? I 0:25 /usr/bin/X11/xdm -daemon
% kill -HUP 198
%
```

Sending a HANGUP signal to xdm tells it to reread the configuration files. All changes are effective immediately after the signal is received.

Interrupting xdm

At certain times, it may be necessary to interrupt xdm. For example, a workstation user might decide to stop using xdm so that prototype servers can be tested. In previous releases, interrupting xdm was difficult and interfered with every xdm session. This is no longer the case.

To interrupt an xdm connection to server *fate:0*, run the following command sequence in Figure 6-8.

Figure 6-8
Interrupting an xdm session

```
% ps aux | grep fate:0
root 6889 1.3 0.1 176 56 r4 S 0:00 grep fate:0
root 6039 0.0 0.1 792 32 ? I 0:01 -fate:0
% kill -TERM 6039
%
```

The X server will be reset and the xdm login window will disappear. If you want to restore the xdm window on *fate:0* later, simply reboot the X server.

Non-XDMCP Servers and the Xservers File

The Xservers file contains an entry for each display that does not use XDMCP. It is preferable that your servers support XDMCP but many servers do not. If some of your servers do not use XDMCP, you will have to add those servers to the Xservers file.

Entries in Xservers contain a display name, followed by the brand of the display, followed by the word "foreign." The display name must be something that can be passed in the `display` option to any X program. An example Xservers file is shown in Figure 6-9.

Figure 6-9
Example Xservers File

```
destiny:0 Sun3/50 foreign
fate:0 Sun3/80 foreign
chance:0 NCD19/R3 foreign
```

Note

In order for xdm to work with non-XDCMP servers, the xdm host must be able to connect to the server on the remote display. Consult your X server documentation for more information on setting access controls in the server. Be sure to add the name of the xdm host to the list of client machines.

After adding a server to the Xservers file, you may also edit xdm-config to specify resources for that server. You may specify any resource discussed earlier in this chapter. Resources for non-XDMCP servers are the same as XMDCP servers. However, there is one special parameter that must be specified.

You must add the following line to the xdm-config file for each non-XDMCP server:

```
DisplayManager.display_0.authorize:false
```

where *display* is the name of that server. Part of XDMCP provides for authorization of the xdm host; you will not be able to use this feature in a non-XDMCP server. Setting the resource to "false" disables this feature.

For example, Figure 6-10 contains the lines in xdm-config for the three servers defined in the previous Xservers file.

Figure 6-10
Example xdm-config File

```
DisplayManager.fate_0.authorize: false
DisplayManager.destiny_0.authorize: false
DisplayManager.chance_0.authorize: false
```

Typical xdm Errors

If there are errors in the configuration files, diagnostic messages are logged to the xdm-errors file. These messages can help you debug your configuration.

Some typical messages include:

```
...access denied, cannot connect to host
```

This means that the xdm host cannot connect to the server. Consult your server documentation and add the xdm host to the access control list.

```
Hung in XOpenDisplay
```

This message indicates that a connection cannot be made to the server. Check to see that the server is on the network; try to ping the machine. A similar message is:

```
Server open failed for <server>:0
```

This also indicates that the server could not be reached.

Invoking xdm

xdm should be invoked at boot time from the local boot file, /etc/rc.local. Figure 6-11 contains the lines you should add to the rc.local file.

Figure 6-11
Invoking xdm

```
#
# Startup X Window display manager.
#
if [ -f /usr/bin/X11/xdm ]; then
 /usr/bin/X11/xdm -daemon
 /bin/echo "Starting xdm." >/dev/console
fi
```

Font Management 7

Users can specify different fonts when displaying menus, text, buttons, etc. of many clients. Fonts are specified using fully-qualified names, such as `-bitstream-charter-bold-i-normal-11-80-100-100-p-68-iso8869-1`, aliases, or wildcards. The X server maintains a database of known font names; when you ask for a particular font, the server tries to find its name in the master list. If the font can be found, it is loaded and displayed.

Fonts are specified in special .snf files and are organized into font directories. The server has a font search path made up of one or more font directories. The server uses this search path to build its internal font name database.

This chapter presents techniques needed to manage fonts. It is assumed that you already have fonts for your X display and have configured your X terminal or workstation so that the server has access to all font directories.

Defining fonts.dir Files

In addition to font files, each font directory contains a file called fonts.dir. fonts.dir files serve as databases for the X server when it searches directories in the default font path. Font directories are located in /usr/lib/X11/fonts.

Each fonts.dir file contains a list of all the font files in the directory with their associated font names in two-column format, as shown in Figure 7-1.

The first line lists the number of entries in the file (that is, the number of fonts in the directory). The first column of the second and subsequent lines lists the font file. The second column lists the actual font name associated with the file.

Figure 7-1
fonts.dir File

```
4
charBI08.snf -bitstream-charter-bold-i-normal--11-80-100-100-p-68-iso8859-1
charBI10.snf -bitstream-charter-bold-i-normal--14-100-100-100-p-68-iso8859-1
charBI12.snf -bitstream-charter-bold-i-normal--17-120-100-100-p-105-iso8859-1
charBI14.snf -bitstream-charter-bold-i-normal--19-140-100-100-p-117-iso8859-1
```

mkfontdir reads the font files in directories in the font path, extracts the font names, and creates a fonts.dir file in each directory. If fonts.dir files already exist on your system, you will not have to use mkfontdir unless you load new fonts or remove existing ones.

Adding Fonts

Use the following procedure to add a new font:

1. Move the new font to the directory you want (for example, /usr/lib/X11/fonts/misc).
2. Type `mkfontdir /usr/lib/X11/fonts/`*dir_name* where *dir_name* is the directory where the new font resides. An entry for the new font is added to fonts.dir.
3. Edit fonts.alias if you want to add an alias for the new font. Refer to "Making Font Aliases" in the following section.
4. Type `xset fp rehash` to make the server reread the font databases and alias files in the current font path.
5. Type `xset fp +`*dir_name* to add the directory to the server's font search path.

Making Font Aliases

You can abbreviate font names by aliasing, that is, associating them with alternative names. You can create a file called fonts.alias in any directory in the font search path. The X server uses both fonts.dir and fonts.alias files to locate fonts in the font path.

Note

When you create or edit a fonts.alias file, the server does not automatically recognize the aliases you have. You must make the server aware of the changes. Refer to "Making the Server Aware of Aliases."

fonts.alias has a two-column layout similar to fonts.dir. The first column contains the alias name. The second column contains the actual font names. If you want to specify an alias that contains spaces, enclose the alias in double quotes. If you want to include double quotes or other special characters, precede each special symbol with a backslash. Figure 7-2 shows an example fonts.alias file:

Figure 7-2
fonts.alias File

```
xterm12   -adobe-courier-medium-r-normal--12-120-75-75-m-70-iso8859-1
xterm14   -adobe-courier-medium-r-normal--14-140-75-75-m-90-iso8859-1
xterm18   -adobe-courier-medium-r-normal--18-180-75-75-m-110-iso8859-1
```

As the names of the aliases suggest, this example file contains aliases for a font with three different point sizes that are easily readable in xterm windows. You can also use wildcards within the font names in the second column of an alias file (as long as wildcards expand unambiguously). The following figure shows a wildcarded fonts.alias file:

Figure 7-3
Wildcarded fonts.alias File

```
xterm12   *courier-medium-r-*-120*
xterm14   *courier-medium-r-*-140*
xterm18   *courier-medium-r-*-180*
```

Making the Server Aware of Aliases

After you create or update an alias file, the server does not automatically recognize the aliases you have. In order to make the server reread the font databases and alias files in the current font path, type `xset fp rehash`.

What Now?

For more details on font management, refer to Chapter 5 of the *X Window System User's Guide* and the mkfontdir(1) and xset(1) man pages.

Error Messages

A

For CXwindows

The following error messages are some of the more common ones you might see.

Message	Action
Xlib: connection to *"mydisp*:0:0" refused by server Xlib: Client is not authorized to connect to Server Error: Can't Open display	Type `xhost +`*myhost* on your server to allow clients access. Refer to xhost(1).
Error: Can't Open display	Set your environment variable: `setenv DISPLAY` *mydisp*`:0`
xfoobar: Command not found	Make sure the client is on your system and /usr/bin/X11 is in your search path.
X Toolkit Warning: Cannot allocate colormap entry for *"foobar"*	Choose a color name the server recognizes. Names are located in the /usr/lib/X11/rgb.txt file. Or if the colormap is full, either select a color already loaded or terminate a client, which will free space in the colormap for the color you want to use.

For CONVEX OSF/Motif

The following error messages are some of the more common ones you might see.

Invalid accelerator specification on line 22 of configuration file

Means the key does not exist. For example,

```
"Restore" _R <Key>F15 f.normalize
```

gives this error message if there are not 15 function keys.

Invalid mnemonic specification on line 25 of configuration file

Means the mnemonic must exist in the string. For example, if the string is "Move," the mnemonic must be _M, _o, _v, or _e. The following definition gives the above error:

```
"Move" _Z Alt<Key>F7 f.move
```

Invalid key specification on line 40 of configuration file

Means the key does not exist. For example,

```
<Key>F15 window|icon f.raise
```

gives this error message if there are not 15 function keys.

mwm: Another window manager is running

Means mwm will not run at the same time as another window manager. You must stop the other window manager before you can start mwm.

Glossary

aliasing — See **font aliasing.**

app-defaults — The system-wide defaults resource file.

awm — A window manager no longer supported by CONVEX and M.I.T. with X11 Release 4.

background window — See **root window.**

bitmap — A grid of pixels, each of which are black, white, or a color (for color displays).

border — See **window frame.**

CONVEX OSF/Motif — The CONVEX implementation of the OSF/Motif graphic user interface.

CXwindows — The CONVEX implementation of M.I.T.'s X Window System, X11 Release 4.

class name — A general component for a set of resources. Class names begin with uppercase letters.

clients — Application programs that perform specific tasks.

control protocol — A mechanism that allows a display to request login service from a remote host.

default value — A function-dependent value supplied when you do not specify one.

display manager — A client (xdm) that manages a collection of remote displays from a single system.

focus — To select a window and make it active by moving the cursor within that window's borders.

font — A style of text characters.

font aliasing — A way to associate font names with abbreviated names.

icons — Graphic symbols that represent inactive windows.

instance name — A specific component for an individual resource and a subset of its class. Instance names begin with lowercase letters.

mouse — An input device, usually with two or three buttons, that when moved across a flat surface, causes the pointer or cursor to move accordingly on your display.

mwm — The Motif Window Manager; a client offered with CONVEX OSF/Motif.

mwmrc — The resource start-up file in your home directory used by mwm that defines buttons, variables, and menus.

OSF — Open Software Foundation.

parameter — A value required by a client to perform its function.

pixels — Abbreviation for picture elements; the dots that form the images that are displayed on your screen.

pointer — The symbol that appears on your display indicating the current location of the cursor.

protocol — A mechanism for communicating between clients.

resource — A parameter that controls behavior or appearance.

root window — A shaded area that covers the entire screen behind all other windows.

scroll area — The area of the scroll bar that contains the slider and scroll arrows.

scroll bar — A graphic device consisting of a slider, scroll area, and scroll arrows. You change your view by moving the slider up or down in the scroll area or by pressing a scroll arrow.

select — See **focus**.

server — A machine that provides display capabilities and keeps track of input.

slider — The part of the scroll bar that is dragged along the scroll area to change your view.

title bar — The bar at the top of the window frame.

twm — The Window Manager; a client offered with CXwindows.

twmrc — The resource start-up file in your home directory used by twm that defines buttons, variables, and menus.

uil — User Interface Language; an OSF/Motif core client.

user preferences — Features that you can customize for an X application using a resource data base.

uwm — A window manager no longer supported by CONVEX and M.I.T. with X11 Release 4.

widget — A graphic device designed to accomplish a specific set of tasks, either individually or in combination with others. Widgets always have a window associated with them.

wildcarding — A way to shorten font or resource names by substituting an asterisk (*) for one or more characters. Wildcards must expand unambiguously.

window — A rectangular region on your root window created by a client.

window frame — The area surrounding a window.

window manager — A client that controls the general operation of your window system and the screen's real estate (for example, mwm).

X Window System — A window environment designed by M.I.T. for networked computer systems.

XDMCP — X Display Manager Control Protocol; see **control protocol**.

Xdefaults — The resource data base file located in your home directory where user preferences are stored.

xdm — X Display Manager (an X client); see **display manager**.

xfontsel — A client used to browse through and select available fonts.

xterm window — An X client that functions as a terminal emulator.

Index

A

adding new fonts 7-2
adding new servers 6-7
additional reading vii
aliasing 7-2
app-defaults 4-1
application defaults, mwm 5-6
application defaults, twm 5-1
associated documents vii
Athena widget examples 1-3
audience of manual v
awm 5-1

B

background window 2-2
bitmap display 2-1

C

class names 4-2
client 2-3
command line options 3-1
command line options, list 3-2
comments viii
components 4-2
configurable defaults 4-1
configuring the X server 4-6
configuring xdm 6-2
contributed clients 1-2
control protocol 6-1
CONVEX clients 1-2
CONVEX OSF/Motif clients 1-4
CONVEX OSF/Motif environment 1-4
core clients, CONVEX OSF/Motif 1-4
core clients, CXwindows 1-1
cursor 2-2
cursor list 2-2
customer service vii
CXwindows clients 1-1
CXwindows unsupported clients 1-2

D

demo programs 1-3
display fonts 3-1
display manager 6-1
display manager, defined 2-5
displays not using XDMCP 6-8
documents, associated vii

E

environment, CONVEX OSF/Motif 1-4

F

focus on a window 2-2
font aliasing 7-2
font database file 7-1
font database file, example 7-2
font management 7-1
font management, more information 7-3
font techniques 7-1
font wildcarding 7-3
fonts.alias file 7-2
fonts.alias file, example 7-3
fonts.dir file 7-1

H

hierarchies, see resources
how to order documents vii

I

icon 2-1
individual options 3-1
instance names 4-2
interrupting xdm 6-8
invoking xdm 6-10

L

libraries 1-3, 1-4
list of cursors 2-2

M

making font aliases 7-2
mkfontdir 7-2
Motif Window Manager 5-6
mouse 2-2
mwm 5-6
mwmrc 5-6

N

notational conventions vi
notes vi

O

O'Reilly & Associates, Inc. examples 1-3
O'Reilly & Associates, Inc. manuals vii
ordering documentation vii
organization of manual v
OSF/Motif libraries 1-4
overview, CONVEX OSF/Motif product 1-4
overview, CXwindows product 1-1

P

pointer 2-2
programming libraries 1-3, 1-4
public-domain clients 1-2
purpose of manual v

Q

questions viii

R

reader comments viii
reconfiguring xdm 6-7
resource manager 4-7
resources 4-1
resources, precedence 4-6
root window 2-2
running clients 2-4

S

screen fonts 3-1
selecting a window 2-2
server 2-3
server-client model 2-3
servers with XDMCP 6-7
setting resources 4-7
specifying display fonts 3-1
specifying fonts 7-1
specifying resources, example 4-4
standard client options 3-1
superuser and xdm 6-1
supported OSF/Motif clients 1-4
supported X clients 1-1
supporting XDMCP servers 6-7
system-wide resources 4-1

T

TAC viii
technical assistance viii
Technical Assistance Center viii
terminal emulator 2-4
twm 5-1
twmrc 5-1
twmrc buttons, default 5-3
twmrc menus, default 5-4
twmrc variables, default 5-2

U

unsupported window managers 5-1
unsupported X clients 1-2
user preferences 4-1
uwm 5-1

W

wildcarding 4-5, 7-3
window environment 2-1
window manager, activities 2-4
window manager, defined 2-2, 2-4
window manager, mwm 5-6
window manager, twm 5-1
windows 2-1

X

X core clients 1-1
X demos 1-3
X Display Manager Control Protocol 6-1
X libraries 1-3
X Window System, defined 2-1
Xdefaults 4-1
xdm and non-XDCMP servers 6-9
xdm configuration file 6-2
xdm daemon 6-1
xdm login window, example 6-3
xdm, defined 2-5, 6-1
xdm, invoking 6-10
xdm, what it does 6-2
xdm-config file 6-2
xdm-config, special parameter 6-9
XDMCP 6-1
xfontsel 3-1
xlsfonts 3-1
xrdb 4-6
Xreset file 6-6
Xservers file 6-8
Xservers file, example 6-8
Xsession 6-5
Xsession file, example 6-6
Xstartup file 6-4
xterm window 2-1

About CONVEX

In 1984, CONVEX Computer Corporation introduced the world's first supercomputer under $1 million. Today, the company leads the industry with its extensive product offering, innovative architecture, advanced system software, availability of third-party applications programs, international distribution, and installed base of supercomputers.

CONVEX markets its products primarily for scientific, engineering, and technical applications in areas such as geophysical research, aerospace simulations, computational chemistry, computer-aided engineering, image processing, and molecular biology. Through CONVEX's supercomputers, users can access leading-edge technology. These powerful systems run computationally intensive software much faster than other systems in the same price range.

CONVEX supercomputers take full advantage of integrated vector and scalar architecture. Multiprocessor versions use a unique technology, Automatic Self Allocating Processor (ASAP). ASAP technology allows multiple jobs to run quickly and efficiently while boosting the performance of a single job. ASAP technology allows other jobs or parts of jobs to run concurrently with a parallel job when processor time is free.

CONVEX supercomputers run an enhanced version of the UNIX operating system. Over 450 third-party software packages take advantage of CONVEX's advanced architecture. Additionally, CONVEX has developed the CONVEX-to-VAX User Environment (COVUE) that enables VAX/VMS users to be productive immediately in the UNIX environment.

Corporate Headquarters
CONVEX Computer Corporation
P.O. Box 833851
Richardson, TX 75083-3851
Phone: 214/497-4000
Fax: 214/497-4848

North America
Canada: 416/940-3644
Eastern USA: 301/345-2400, 2401
Western USA: 408/453-5700

Far East
Headquarters: 65-733-4355
Australia: 61-7-221-2404
Japan: 81-3-221-5105

Europe
Central Region: 49-69-6668081
Northern Region: 31-30-888368
Southern Region: 33-1-30589300
Western Region: 44-372-386696

CONVEX

The trend in supercomputing is toward visualization of data using software that transforms numerical computational output into geometric images. CONVEX OSF/Motif and CXwindows allow workstation users to easily access the speed and power of CONVEX supercomputers.

M.I.T.'s X Window System and the Open Software Foundation's OSF/Motif graphical user interface (GUI) are fast becoming industry-standard products. The X protocol effectively translates applications running on diverse hardware platforms and renders those applications in windows on a bitmapped-display device such as a workstation. OSF/Motif defines how programs appear to the user—providing a common "look and feel" from one display device to another and from one application to another.

CONVEX OSF/Motif and CXwindows bring supercomputing and graphics tools to your desktop to provide seamless network integration. And because many major companies support X and OSF/Motif, these CONVEX tools are the basis for integrating software across a range of computers.

These current user interfaces are building blocks for future graphics and visualization products, all of which will rely on de facto or emerging standards (such as the X Window System, the Programmer's Hierarchical Interactive Graphics Systems (PHIGS+), and the PHIGS+ Extension to X (PEX) that provides protocols for rendering three-dimensional objects). CONVEX plans to support PEX soon after its release by M.I.T.