The X Primer

First Edition

CONVEX COMPUTER CORPORATION

CONVEX Computer Corporation
3000 Waterview Parkway
P.O. Box 833851
Richardson, TX 75083-3851
United States of America
(214)497-4000

The X Primer

Order No. DSW-218

First Edition
April 1992

CONVEX Press
Richardson, Texas
United States of America

The X Primer

Order No. DSW-218

Printed in the United States of America

Revision information for

The X Primer

Edition	Document No.	Description
First	710-022630-000	April 1992. Initial release.

Contents

Figures

Using this book

Purpose and audience

The X Primer is a guide for new users of the X Window System using version 11 release 4 or 5 (X11 R4 and R5). This guide includes:

- How to use the X Window System with the Motif Window Manager (mwm)
- How to set up your X defaults
- How to customize the Motif Window Manager
- How to select fonts
- How to write X start-up files
- How to secure your X server

This guide addresses users who are new to the X Window System but are familiar with common UNIX commands.

Organization

This guide is organized into the following chapters and appendixes:

- Chapter 1 introduces general concepts necessary to start using the X Window system.
- Chapter 2 describes how to start X and interact with the Motif Window Manager.
- Chapter 3 describes how to set defaults that affect the appearance and behavior of X-based programs.
- Chapter 4 describes how to customize the Motif Window Manager to modify and create new menus, change mouse button behavior, and change key bindings.
- Chapter 5 describes how to select and specify fonts.
- Chapter 6 describes how to write an X start-up file.

- Chapter 7 describes how to secure your X server so that other users cannot access it.
- The Glossary contains definitions of X Window System terms.

Notational conventions

This section discusses notational conventions used in this book.

Command syntax

Consider this example:

```
COMMAND input_file [...] {a | b} [output_file]
   ①        ②       ③     ④          ⑤
```

1. `COMMAND` must be typed as it appears.
2. *input_file* indicates a file name that must be supplied by the user.
3. The horizontal ellipsis in brackets indicates that additional input file names may be supplied.
4. Either `a` or `b` must be supplied.
5. [*output_file*] indicates an optional file name.

General conventions

In general, the following conventions are used in this guide:

- Italics:
  - Designate user-supplied variables in a command-line example
  - Introduce new and important terms
  - Identify variables in mathematical equations
  - Indicate document titles
- `Constant-width font` designates input and output, including:
  - Command names and options
  - Display examples, printout examples, and error messages returned
  - X resource specifications
- **`Bold constant-width font`** designates command line input that you should enter as you read.

- Horizontal ellipsis (...) shows repetition of the preceding item(s).
- Vertical ellipsis shows that lines of code have been left out of an example.
- Words and abbreviations that indicate keyboard keys you press are identified in a distinctive bold type. For example, **RETURN** refers to the carriage return key. Words separated by a hyphen indicate two keys that you must press simultaneously. For example, **CTRL-X** indicates that you must press and hold down the **CTRL** key and then press the **X** key.
- The word "enter" in a phrase such as "enter **`ls`**" means that you type the command and then press **RETURN**.
- The ConvexOS prompt is printed as a percent sign (`%`).

Note

A note highlights supplemental information.

Associated documents

Using this software may require information not specific to the tasks described in this document.

For more detailed information on the X Window System, you can find *Volume 3, X Window System User's Guide* from O'Reilly & Associates, Inc., at your local computer book store.

For more information on the ConvexOS operating system, you can order these books from CONVEX Computer Corporation:

- *ConvexOS Primer* (DSW-133) has basic self-instruction for learning and using the ConvexOS operating system.
- *ConvexOS Man Pages for Programmers* (DSW-332) contains copies of Sections 2 through 5 of the online man pages. These man pages are primarily concerned with operating system information for programmers.
- *ConvexOS Man Pages for Users* (DSW-331) contains copies of Sections 1 and 7 of the online man pages. These man pages are primarily concerned with operating system information for users.

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, as listed on the front cover.

Technical assistance

If you have questions that are not answered in this book, contact the CONVEX Technical Assistance Center (TAC).

- Within the continental U.S., call 1(800)952-0379.
- From Canada, call 1(800)345-2384
- Outside continental U.S., contact local CONVEX office.

Reader response

If you have comments or questions about the contents of this book, please notify the CONVEX documentation department by using the Reader's Forum at the end of this document.

The `contact` utility

The TAC recommends using the `contact` utility to report a hardware, software, or documentation problem. The `contact` utility is an interactive program that helps the TAC track reports and route them to the CONVEX personnel most qualified to fix a problem.

After you invoke `contact`, it prompts you for information about the problem. When you finish your report, `contact` mails it to the TAC electronically. The TAC notifies you within 48 hours that your report has been received.

To use `contact` requires:

- UNIX-to-UNIX Communication Protocol (UUCP) connection to the TAC.
- Full path name of the program or utility in question.
- Version number of the program or utility in question.

Refer to the contact(1) man page for complete details.

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. In particular:

- The CXwindows developement team: Martin Streicher, Greg Stiehl, and Tracy Webb.
- The CXwindows test team: Brent Henderson and Tim Powell.
- Technical Assistance Center team: Mark Botner and Keith Towles.
- Software Environments Documentation management: Kathy Harris.
- Software Environments Documentation staff: Ray Cetrone, Dan Connolly, Ken Harward, Henry Smith, and Neal Johnston.
- Editorial Services: Larry Bonura, Clare Bernier, Peggy Gilloon, Sheri Roloff, Robin Sowton.
- Document testing volunteers: Brad Bass, Bruce Wolfe, Jane Caban, David Trawick, Evan Roberts, Tom Finn, Frank Fornara, Joe Machado, Don May, Rob Spray, Steve Whiteside, and Jeff Thompson.
- Document review volunteers: Sean Stroud, Ed Pennington, John Benavides, Jack Sugrue, Mike Sydow, Wayne Greaves, and Linda-Lou Holden.

—Keith Knox

What is X?

1

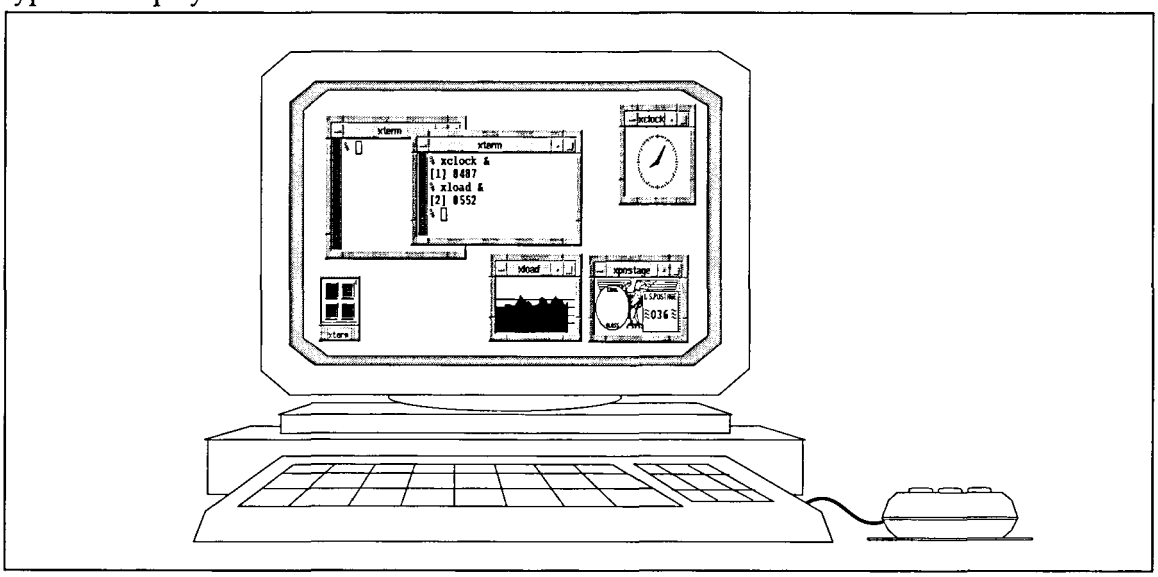
The X Window System, also known as X, is a graphic windowing system designed for networked computer systems with bitmap displays. It was designed at the Massachusetts Institute of Technology (M.I.T.) and first released in 1984.

X divides your screen into areas called windows. These windows represent application programs. A pointer device (usually a mouse) directs keyboard input to the windows.

X enables application programs (called X clients) to run on a remote system and display the output on and receive input from your graphic display device. This means that you can have several windows on your screen, each of which may be an X client running on a different networked computer.

Figure 1 shows a typical graphic display device that has a bitmap display, a keyboard, and a mouse.

Figure 1
Typical X display



X display devices

Graphic display devices fall into two categories: workstations and X terminals. All graphic display devices that are used with X are referred to as X servers.

A workstation is a standalone computer that has its own processor, memory, and possibly a disk drive. To X, a workstation is a computer in the network that has input and display capabilities. A workstation becomes an X server when it is running the X server program.

An X terminal is not a standalone computer. An X terminal is a graphic input and output device that must be connected to a host computer. The X server program is built into the X terminal.

The X display

X displays windows on bitmap displays. A bitmap display is a screen that has a grid of picture elements (pixels) that are black or white or another color if the screen can display color.

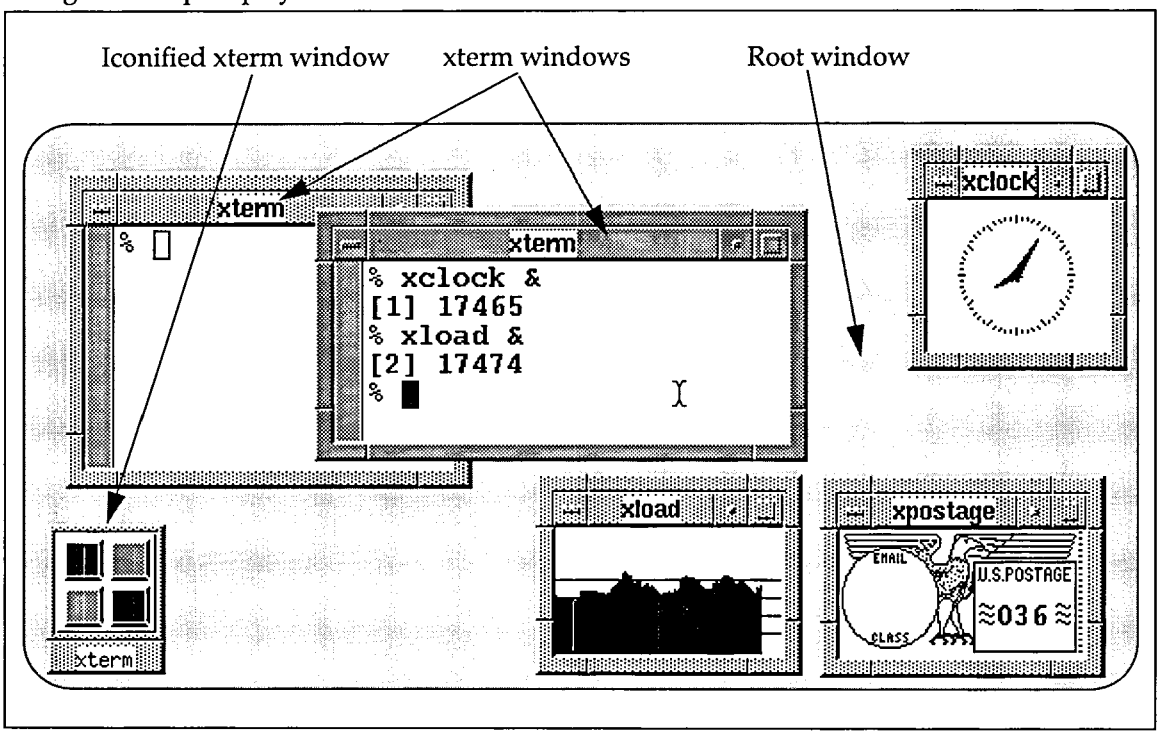
Figure 2 is an enlargement of the bitmap display shown in Figure 1. The windows labeled xterm are terminal emulators. These windows are separate terminal sessions. Once you move the mouse pointer into an xterm (and press the left mouse button on some systems), you can type input to the computer that the xterm is connected to.

The small window in the lower left corner is an iconified xterm window. An icon represents a window that is temporarily closed from input. You might want to turn a window into an icon to make room for more windows on your screen.

The xload window shows the system load on a remote computer, the xclock window shows the time, and the xpostage window shows the number of email messages that this user has.

The grey area behind the windows is called the root window. You can think of the root window as the screen of your terminal. Other windows always appear on top of the root window.

Figure 2
Enlarged bitmap display



What is the client-server model?

X is often talked about in terms of the client and the server. An X client is any program that talks to an X display. For example, xclock, xterm, and xload are all X clients that can be running locally on your workstation or remotely on a computer in the network. In short, anything that creates windows on your display is an X client.

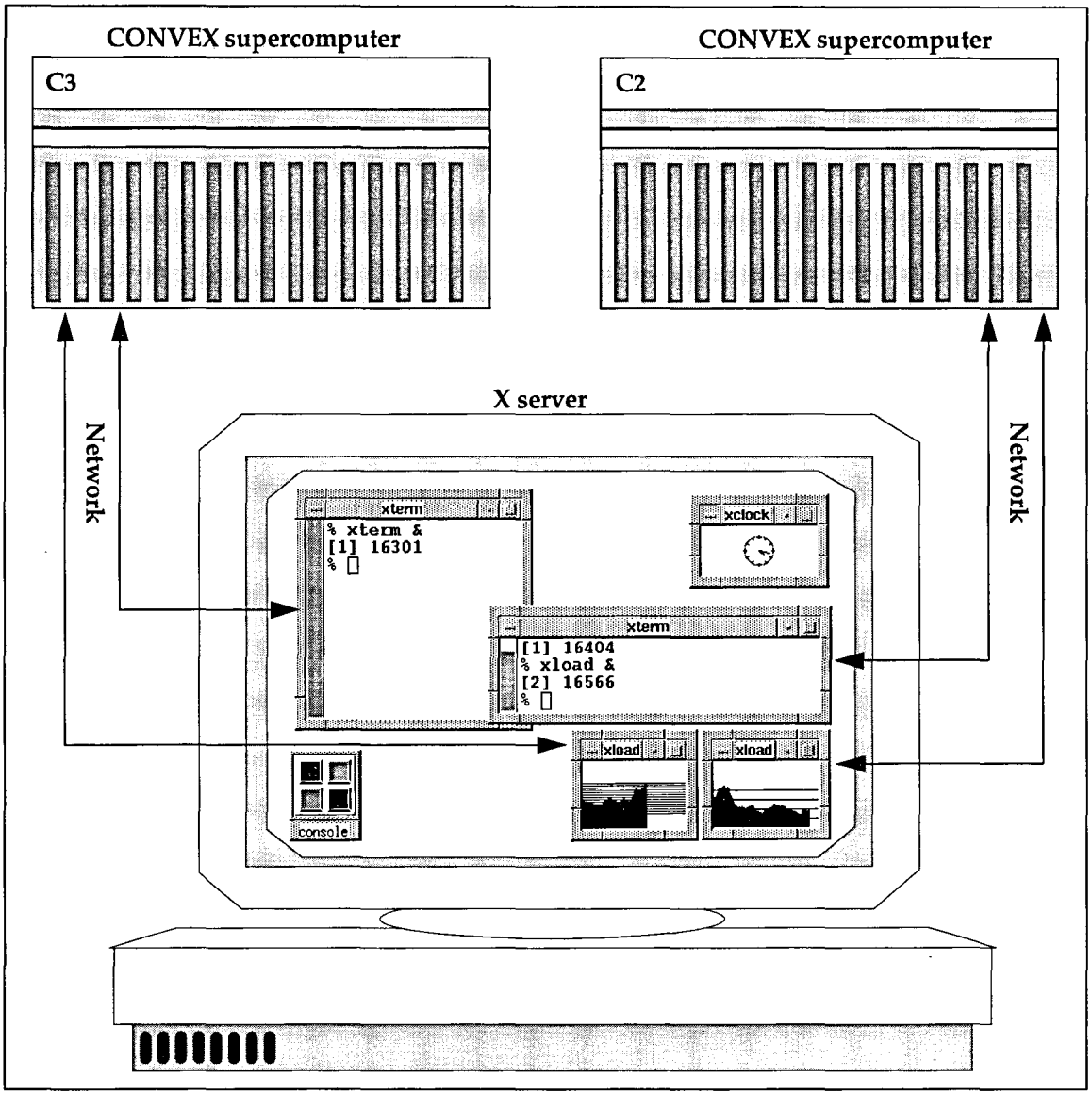
The X server is a program that runs on your display device and provides display and input capabilities for X clients. Your display device is usually referred to as the X server when it is running the X server program.

For example, you might use a workstation as an X server to interact with X clients running on a CONVEX supercomputer. Even though the X client is actually running on the supercomputer, all user input and output occur on the workstation and are communicated across the network using the X protocol. The X protocol is the mechanism that X clients use to speak to X servers.

Figure 3 shows a simple X client-server network. The workstation is displaying six X clients:

- One xterm window and one xload window are running on the CONVEX C3 Series computer.
- One xterm window and one xload window are running on the CONVEX C2 Series computer.
- The xclock window and iconified console window are running locally on the workstation.

Figure 3
Simple client-server network



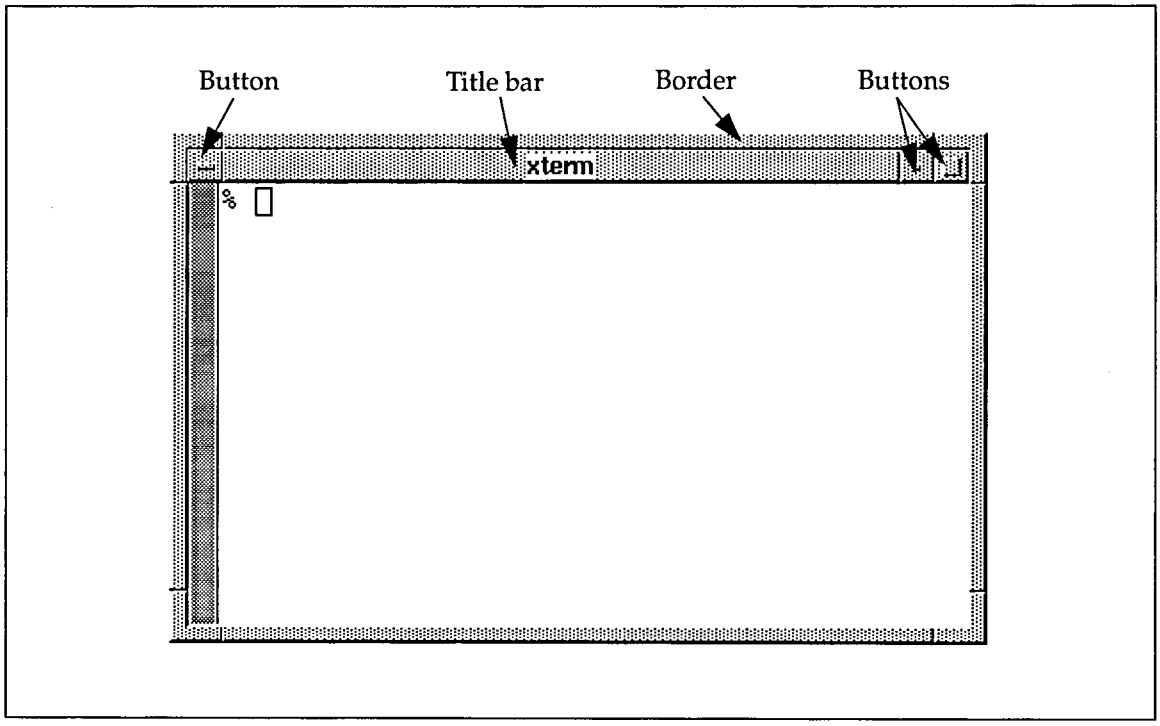
What is a window manager?

A window manager is an X client that enables you to control the size and position of windows on your screen. For example, you interact with the window manager to move, resize, or iconify windows. Only one window manager can run on your X server at a time.

The window manager gives a certain "look and feel" to your X display. For example, the window manager determines the type and functionality of window decorations (borders and title bars) as well as the functions of your mouse buttons.

While many window managers are available, this book only discusses the Motif Window Manager (mwm). Motif is the name of a user interface style designed by the Open Software Foundation and is used in mwm. Figure 4 shows mwm's 3-D window decorations.

Figure 4
Motif window decorations



Using X

2

This chapter gets you started using X and discusses:

- How to log in and start X from either a workstation or an X terminal
- How to start mwm
- How to manipulate windows
- How to use mwm menus
- How to start X clients
- How to use the xterm terminal emulator

Logging in and starting X

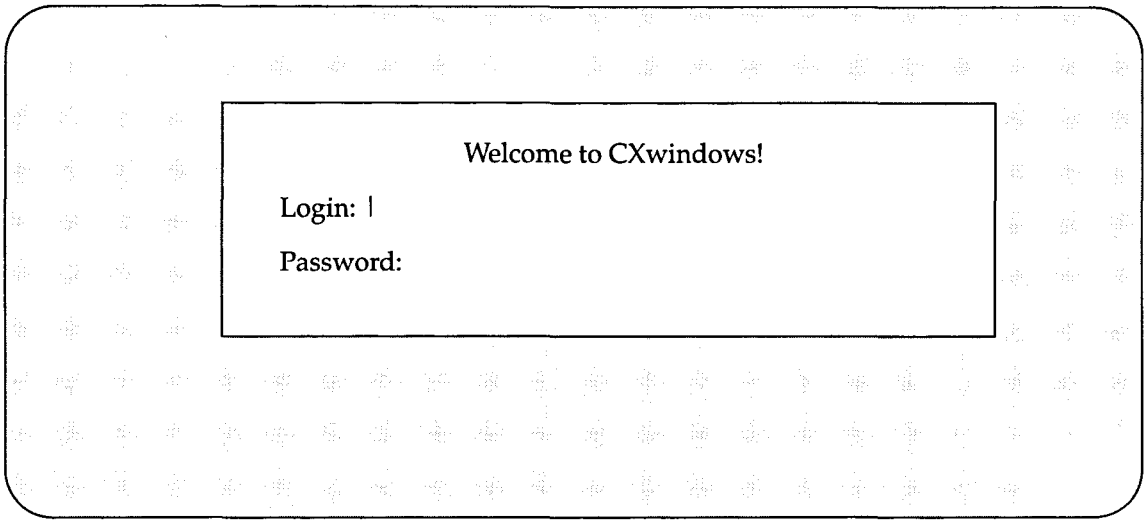
Once you have an X display device, a user account, and a password, then you can log in and start using X. This book assumes that you are either running X from a workstation or an X terminal.

Before you start X, make sure that /usr/bin/X11 is in your path. You can check your path by entering `echo $PATH` at the command line. Your path is usually set in your .cshrc or other shell configuration file.

Logging in from the xdm login window

If a window similar to the window shown in Figure 5 is on your screen, then your system is running the X Display Manager (xdm). If you are logging in on a workstation and xdm is not running, skip to the section called "Starting X on a workstation without xdm."

Figure 5
The xdm login window



xdm is managed by your system manager. xdm is helpful because it makes it easier to log in and start using X. It also provides important security features. For more information on X security, refer to Chapter 7, " Securing your X server."

If the xdm login window shown in Figure 5 is on your screen, then X is already running; log in at the prompt. Type in your login ID and press **RETURN**. Then type in your password and press **RETURN**.

When you log in from the xdm window, xdm looks for a file called .xsession in your home directory. The .xsession file specifies the window manager and other X clients that you want X to start automatically when you log in. If no .xsession file exists in your home directory, xdm creates a default .xsession file in your home directory that creates one xterm window and starts a window manager.

If the border around your xterm window has the characteristic mwm decorations, then the mwm window manager is running. If another window manager is running and you wish to use mwm, refer to Chapter 6, " Writing an X start-up script" to learn how to specify the mwm window manager in your .xsession file.

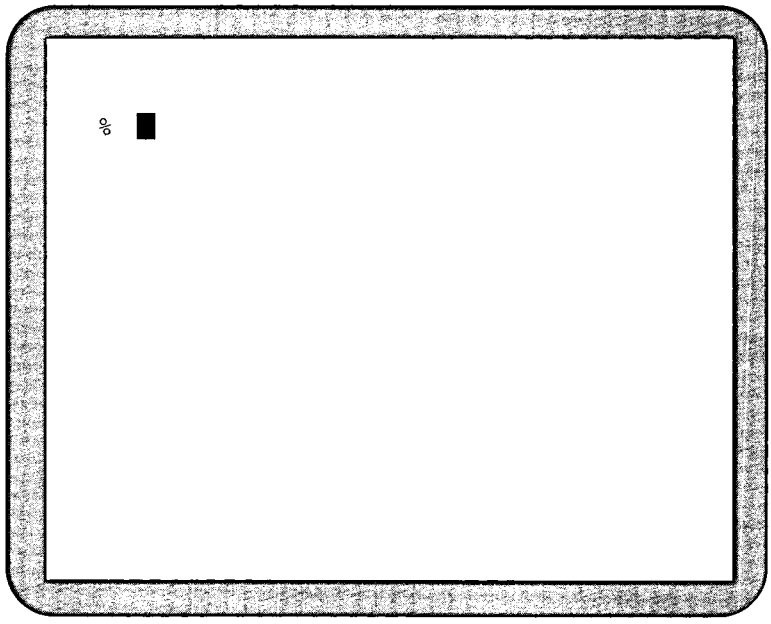
Starting X on an X terminal without xdm

We recommend using xdm particularly when using an X terminal. If you decide to use an X terminal without xdm, then you should consult your X terminal's documentation to determine how to start X.

Starting X on a workstation without xdm

Log in to your workstation. If your entire workstation screen is functioning as a single terminal as shown in Figure 6, then you must start X manually.

Figure 6
Workstation not yet running X



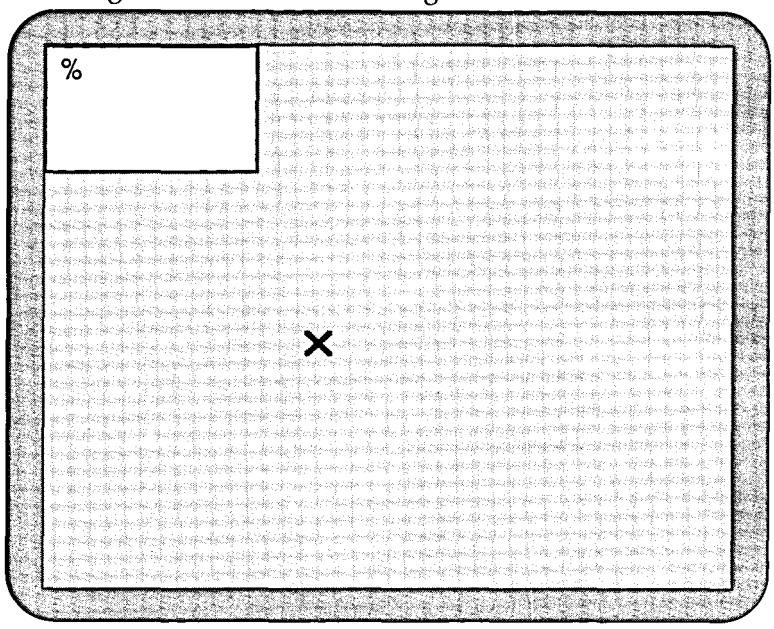
On many workstations, you can start X by entering `xinit` at the command line. `xinit` starts X and looks for a file called .xinitrc in your home directory. On many workstations, the .xinitrc file specifies the window manager and other X clients that you want automatically started when you issue the `xinit` command.

If `xinit` does not work, consult your workstation's documentation on how to start X.

If you start X and a single xterm window appears in the upper left corner of your screen with no border around it as shown in Figure 7, then you do not have an X start-up file, and mwm is not yet running. You can start mwm by moving the mouse pointer into the xterm window and entering `mwm &` at the command line. The ampersand (`&`) places mwm in the background so that you can type other commands in the window.

To write an X start-up file, refer to Chapter 6, " Writing an X start-up script."

Figure 7
X running without a window manager



Using X with mwm

After you start X and mwm, you can start manipulating the windows on your display. The information presented here is based on mwm defaults. If your X setup has already been customized, you can revert back to the defaults by renaming or removing your .mwmrc, .xsession, .Xresources, and .Xdefaults files.

Choosing the input focus window

Selecting a window for keyboard input is called *focusing*. The window that has the input focus is called the active window. A *focus policy* refers to the way you select a window for input focus.

With mwm, you can choose between two focus policies:

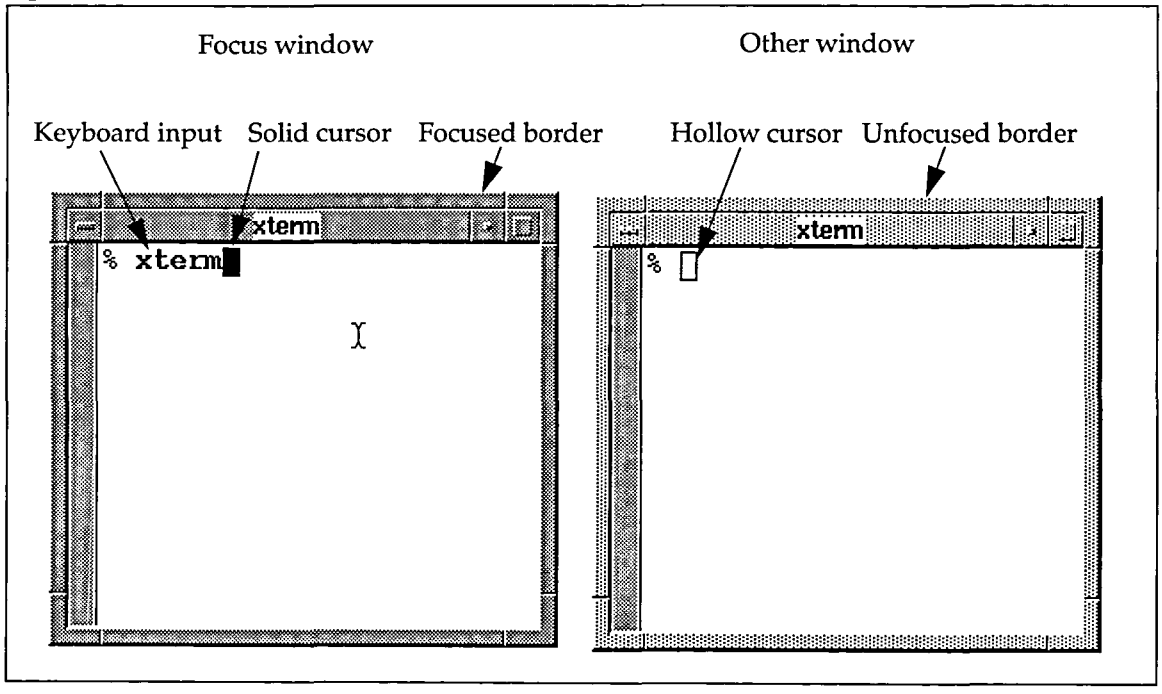
- **Implicit (or pointer) focus policy**—Enables you to select the input window by moving the mouse pointer into a window.
- **Explicit (or click to type) focus policy**—Enables you to select a window for input by moving the mouse pointer into a window and clicking the left mouse button.

If you are running CXwindows (the CONVEX version of X), implicit focus policy is the default. Most other versions of mwm use the explicit focus policy by default. You can change the focus policy through an mwm resource setting as described in Chapter 4, " Customizing the mwm window manager." This book assumes that you are using the implicit (pointer) focus policy.

As shown in Figure 8, several things happen when a window becomes active:

- The window's border changes color or is highlighted.
- If the window has a text cursor, it may become solid (as with xterm).
- The mouse pointer changes (usually from X to I).
- Keyboard input is directed to that window.

Figure 8
Input focus window



Using the mouse

The mouse pointer directs keyboard input and mouse button input. Most pointer devices have three buttons. By moving the pointer to some specific location and performing a button action, you can invoke an mwm command. You can perform six basic actions with the mouse:

- **Press**—Hold down a mouse button.
- **Release**—Stop pressing a mouse button.
- **Click**—Press and release a mouse button without pausing.
- **Double-click**—Click the mouse button twice without pausing.
- **Triple-click**—Click the mouse button three times without pausing.
- **Drag**—Press a mouse button and move the mouse; then release the mouse button.

These terms are used throughout this book.

Manipulating windows

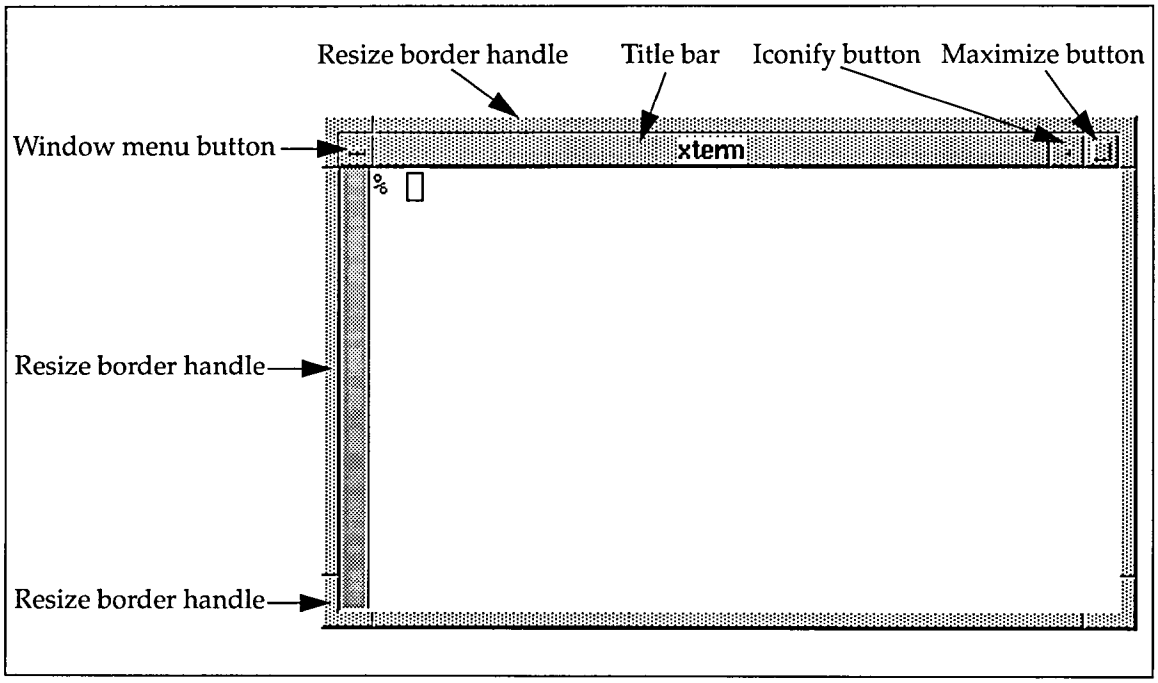
This section explains how to manipulate windows with the mouse. You can invoke window manipulation commands by using the mouse buttons while the mouse pointer is at a particular place on the window. The commands in this section are based on the default mouse button commands. (Refer to Chapter 4, " Customizing the mwm window manager," to customize your mouse button commands.)

If you wish to try the instructions that follow, you may want to have another xterm window on your screen. To create another xterm window, enter **`xterm &`** in your existing xterm window.

Figure 9 shows an xterm window with its mwm border labeled. The parts are:

- **Resize border handle**—Allows you to stretch the borders of the window to resize the window.
- **Window menu button**—Displays the window menu.
- **Title bar**—Contains the window's title and enables the mouse to move the window.
- **Iconify button**—Turns the window into an icon.
- **Maximize button**—Enlarges the window to its maximum size.

Figure 9
mwm window border

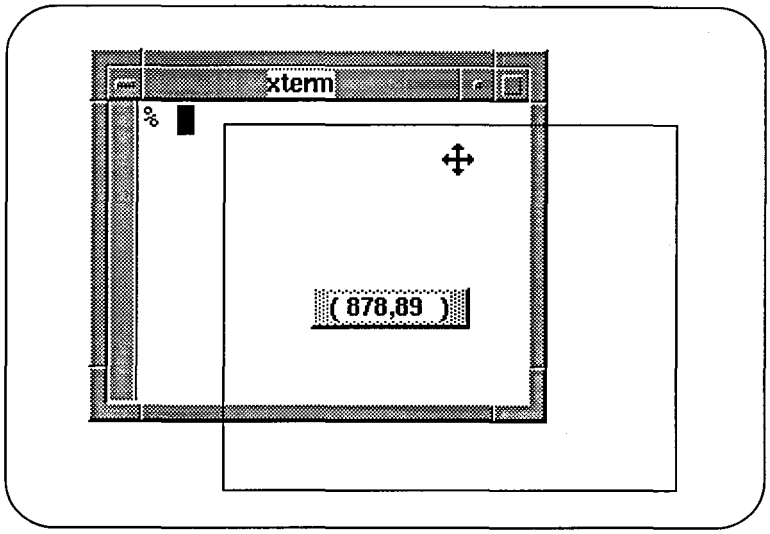


Moving a window

To move a window:

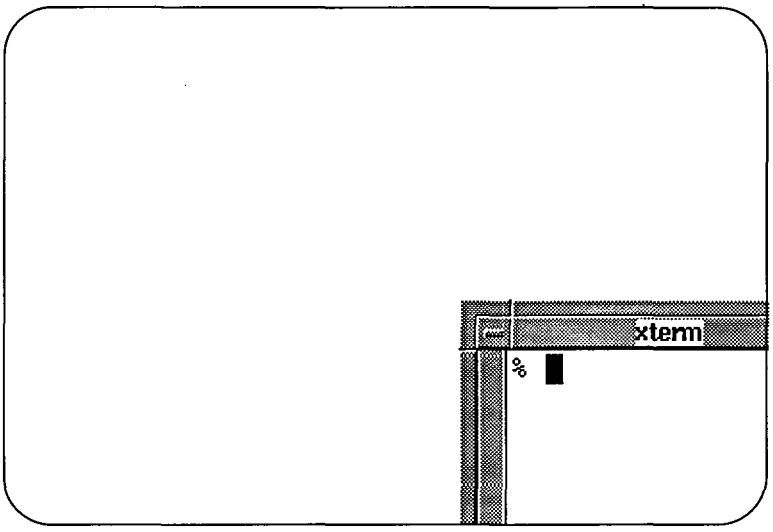
1. Place the pointer on the title bar; the mouse pointer becomes an arrow.
2. Press the left mouse button.
3. Drag the mouse to move the window to a new place. As shown in Figure 10, the mouse pointer changes to a cross arrow pointer, a window outline appears, and a small rectangular box appears that shows the window's X and Y coordinates.
4. Release the button when the outline is where you want to place the window.

Figure 10
Moving a window



One interesting feature of mwm is that you can move a window so that it is partially off the screen as shown in Figure 11.

Figure 11
A partially hidden window



Raising a window

If a window is partially covered by another window, you can raise the covered window so that no other windows obscure it:

1. Move the mouse pointer to any part of the border of the partially covered window. The pointer changes to a resize arrow.
2. Click the left mouse button. The window rises to the top.

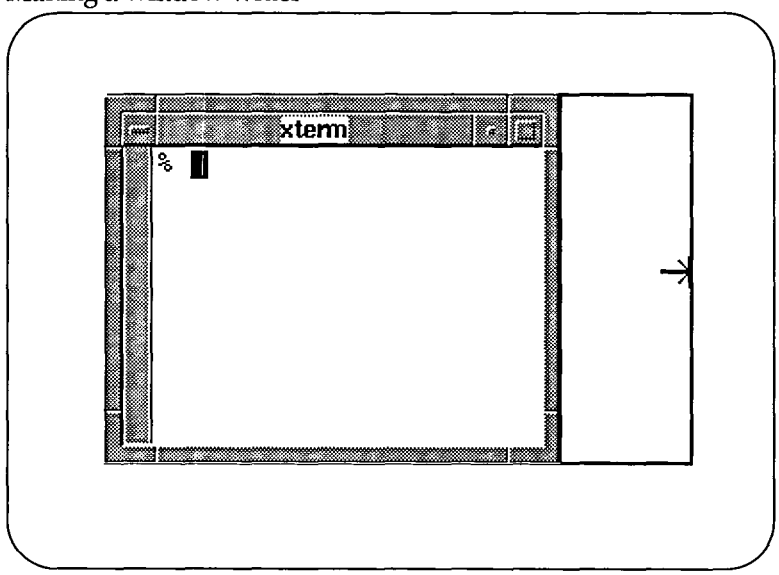
To raise a window that is completely covered by another window, use the Lower option of the window menu to lower the window that is covering it. The window menu is described later in this chapter in "Using the window menu" on page 20.

Changing the size of a window

To change the width of a window:

1. Move the mouse pointer to the left or right border. The pointer becomes an arrow pointing at a vertical bar.
2. Press the left mouse button.
3. Drag the border to the left or right. As shown in Figure 12, an outline of the window appears.
4. Release the mouse button when the window is the desired width.

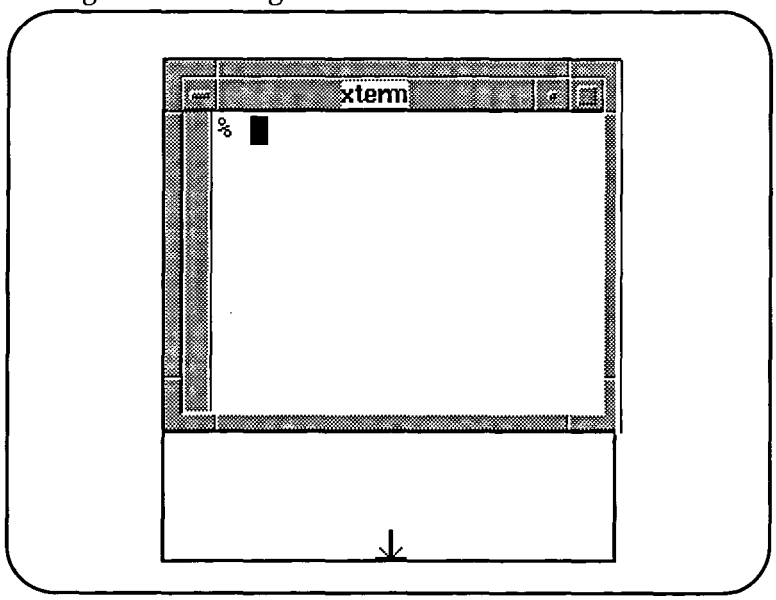
Figure 12
Making a window wider



To change the length of a window:

1. Move the mouse pointer to the top or bottom border. The pointer becomes an arrow pointing at a horizontal bar.
2. Press the left mouse button.
3. Drag the border up or down. As shown in Figure 13, an outline of the window appears.
4. Release the mouse button when the window is the desired length.

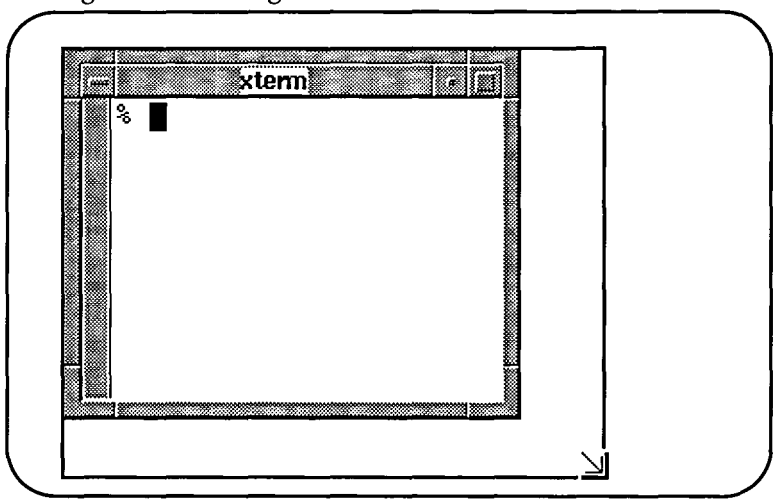
Figure 13
Making a window longer



To change the length and width of a window:

1. Move the mouse pointer to one of the border corners. The pointer becomes an arrow pointing into a corner.
2. Press the left mouse button.
3. Drag the border diagonally to make the window larger or smaller. As shown in Figure 14, an outline of the window appears, and you can change both dimensions of the window at the same time.
4. Release the mouse button when the window is the desired size.

Figure 14
Making a window longer and wider



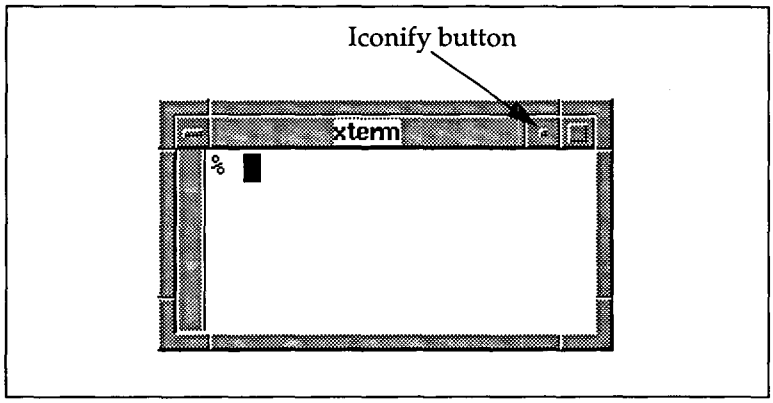
Iconifying a window

An icon is a very small window that represents a larger window. You iconify a window when you want the window temporarily out of the way. To iconify a window:

1. Move the mouse pointer to the Iconify button beside the title bar of the window as shown in Figure 15. The pointer changes to an arrow.
2. Press the left mouse button. The window becomes iconified and is represented by a smaller window that displays the same title.

An iconified window is still active because it can still receive output, but you cannot see the output until the window is de-iconified.

Figure 15
Iconify button



Turning an icon into a window

Double-click the left mouse button on the icon. The icon disappears and the window appears in the place it was before it was iconified.

You can also de-iconify a window by selecting the Restore option from the window menu as discussed in "Using the window menu" on page 20.

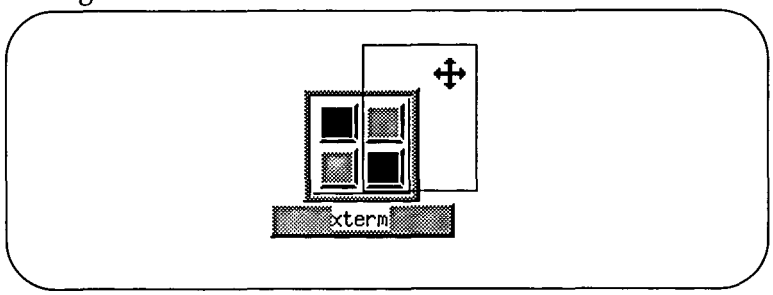
Moving an icon

You move an icon the same way you move a regular window.

1. Place the pointer on the icon; the mouse pointer becomes an arrow.
2. Press the left mouse button.

3. Drag the mouse to move the icon to a new place. As shown in Figure 16, the mouse pointer changes to a cross arrow pointer, and an icon outline appears.
4. Release the mouse button when the outline is where you want to place the icon.

Figure 16
Moving an icon



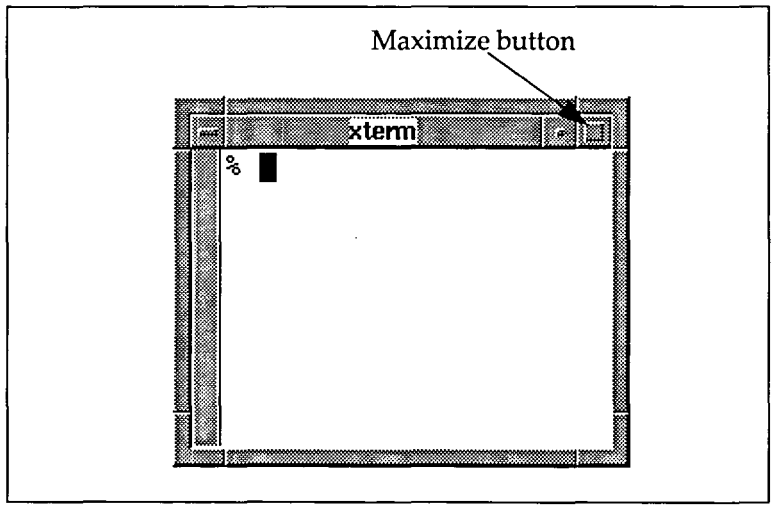
Maximizing a window

Maximizing a window makes a window as large as your entire screen. To maximize a window:

1. Place the mouse pointer over the Maximize button as shown in Figure 17. The pointer becomes an arrow.
2. Click the left mouse button. The window now fills your entire display.

To restore a maximized window to its previous size, click the Maximize button again.

Figure 17
Maximize button



Using the window menu

The window menu provides an alternate way of issuing commands that you usually invoke using one of the mwm border features such as moving or resizing the window. Figure 18 shows the window menu. The options include:

- **Restore**—Turns the icon into a window or returns the maximized window to its regular size.
- **Move**—Enables you to move the window with the mouse or the arrow keys.
- **Size**—Enables you to resize the window with the mouse or the arrow keys.
- **Minimize**—Turns a window into an icon.
- **Maximize**—Increases the size of the window to the size of your screen.
- **Lower**—Places the top window at the bottom of a stack of windows.
- **Close**—Stops the client and destroys the window.

Shaded options are not available. For example, the Minimize option is shaded when a window is iconified as shown in Figure 18.

Using the mouse

To display the window menu with the mouse:

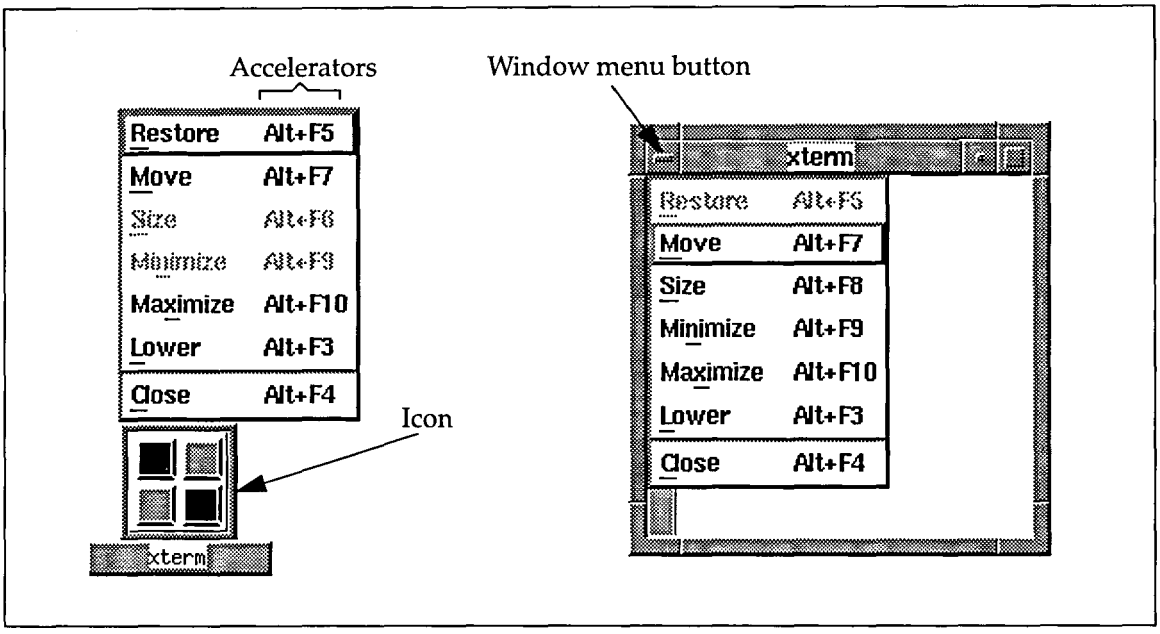
1. Place the mouse pointer over the window menu button on the left side of the title bar. The pointer becomes an arrow.
2. Click the left mouse button. The window menu appears.

You can also display a window menu from an icon by placing the mouse pointer over the icon and clicking the left mouse button.

To select an option from the window menu, click on the desired option with the left mouse button.

You can also select an option while you are opening the menu by continuing to press the left mouse button and moving the mouse pointer over an option. The option becomes boxed. Release the mouse button to select the option.

Figure 18
Window menu



To select an option from the window menu, click on the desired option with the left mouse button.

You can also select an option while you are opening the menu by continuing to press the left mouse button and moving the mouse pointer over an option. The option becomes boxed. Release the mouse button to select the option.

Using keyboard shortcuts

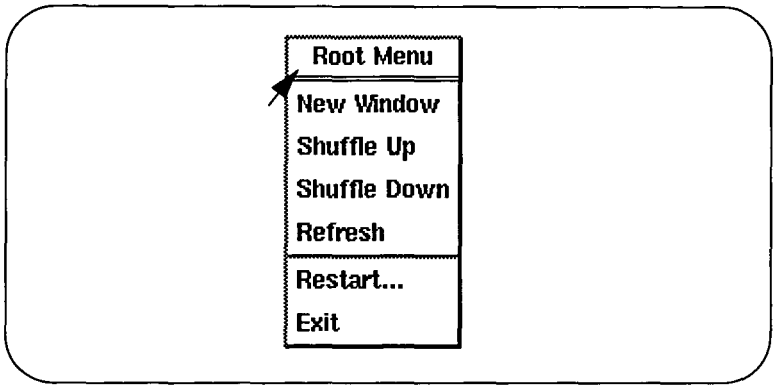
To display the window menu with the keyboard, hold down the SHIFT key and press the ESC or META key. You can select one of the options by using the underlined mnemonic abbreviation or by using the *accelerator* to the right of the option as shown in Figure 18. An accelerator is just a keyboard shortcut. You can use upper- or lowercase when using a mnemonic abbreviation.

For example, from the window menu, you can turn an icon into a window by either typing the character `r` or by holding down the ALT key and pressing the F5 key.

Using the Root Menu

The Root Menu is mwm's main menu. The options on the Root Menu affect your entire display. Display the Root Menu by moving the mouse pointer to the root window and pressing the left mouse button. The root window is any part of your display that is not covered by a window. Figure 19 shows the default Root Menu.

Figure 19
The mwm Root Menu



To select a command from the Root Menu:

1. Press and hold down the left mouse button over the root window. The Root Menu appears.
2. Move the mouse pointer over an option. The option becomes boxed.
3. Release the mouse button to select the option.

The default Root Menu options include:

- **New Window**—Creates a new xterm window on your screen.
- **Shuffle Up**—Moves the bottom window in a stack of windows to the top.
- **Shuffle Down**—Moves the top window in a stack of windows to the bottom.
- **Refresh**—Refreshes your entire X display. This option is useful when your root window displays system messages that detract from the appearance of the root window.

- **Restart**—Stops and restarts mwm. The Restart option is useful for triggering changes that you have made in your .mwmrc file as described in Chapter 4, " Customizing the mwm window manager."
- **Exit**—Exits mwm and can exit X if the last line in your X start-up file starts mwm. For more information, refer to Chapter 6, " Writing an X start-up script."

Using X clients

X clients are application programs that run on an X display. You can start an X client by entering the client's name on an xterm's command line. You should follow the client's name with an ampersand (`&`) to run the client in the background. Figure 20 shows the oclock client being started.

Figure 20
Starting a client

```
% oclock &
[1] 4606
```

Starting an X client on a different computer

You can run an X client on a different computer from the one that is running your xterm window by using the remote shell command, `rsh`. This is useful when you need to run a client whose data exists on another computer or when you wish to run clients on other computers to reduce the system load on a particular computer.

For example, if your xterm window is running on a computer named tornado, you can run xload on a computer named hurricane and display it on an X display named junkfood by following the syntax in Figure 21.

Figure 21
Using `rsh`

```
% rsh hurricane 'xload -display junkfood:0' &
[2] 4607
```

The `-display` option tells the client which display to connect to (usually your own). In Figure 21, it connects to the device called junkfood. The `:0` is always required when you specify a display name; it tells the client which display server to connect to. Because most X display devices only have one display server, this number is usually 0.

If your `rsh` command does not work, check to make sure that you are specifying your X display's name correctly with the `-display` option. To find out what the name of your display is, enter the command **`xdpyinfo`** at the command line as shown in Figure 22. Your display name appears on the first line of `xdpyinfo`'s output.

Figure 22
`xdpyinfo` command

```
% xdpyinfo
name of display: junkfood:0.0
version number: 11.0
vendor string: MIT X Consortium
vendor release number: 5000
maximum request size: 262140 bytes
motion buffer size: 256
bitmap unit, bit order, padding: 32, MSBFirst, 32
image byte order: MSBFirst
number of supported pixmap formats: 1
supported pixmap formats:
 depth 1, bits_per_pixel 1, scanline_pad 32
keycode range: minimum 8, maximum 132
focus: window 0x1c0000d, revert to Parent
number of extensions: 4
 SHAPE
 MIT-SHM
 Multi-Buffering
 MIT-SUNDRY-NONSTANDARD
default screen number: 0
number of screens: 1

screen #0:
 dimensions: 1600x1280 pixels (451x361 millimeters)
 resolution: 90x90 dots per inch
 depths (1): 1
 root window id: 0x23
 depth of root window: 1 plane
 number of colormaps: minimum 1, maximum 1
 default colormap: 0x21
 default number of colormap cells: 2
.
.
.
```

Standard command line options

Most X clients share a common set of command line options. Most of these options control the appearance of the client. This section presents an overview of the standard X client options.

X clients can also have individual options that control the features of the client. For example, the xclock client has an option to run in analog or digital mode. Refer to an X client's man page for a complete list of available options.

There are numerous standard options that most X clients recognize. The following list briefly describes each one:

- `-background`—Specifies the background color of the window. You can also use the `-bg` flag to specify background color.
- `-bordercolor`—Specifies the border color of the application window. You can abbreviate this option as `-bd`.
- `-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. You can abbreviate this option as `-fn`.
- `-foreground`—Specifies the foreground (drawing or text) color of the client. You can abbreviate this option as `-fg`.
- `-geometry`—Specifies the preferred size and position of the client window. For information on specifying geometry coordinates, refer to the mwm man page.
- `-help`—Indicates that a brief summary of the allowed options should be printed to standard error.
- `-iconic`—Specifies that the application should start in iconified form. An application will not start in iconified 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. For more information, refer to the section "Using client names in resource specifications" on page 39.
- `-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.
- `-xrm`—Specifies a resource to load into the resource manager. The argument to this option is a quoted string similar to the lines found in a resource file. For more information, refer to the section "Setting resources from the command line" on page 41.

Using xterm, the terminal emulator

xterm is a terminal emulation program that creates a window that acts as a single terminal. An xterm runs a shell session on a networked computer. You can use an xterm to issue UNIX commands, run programs, or start other X clients. You can also have more than one xterm window on your display; one xterm may be running locally on your workstation and several more may be running on remote computers in the network.

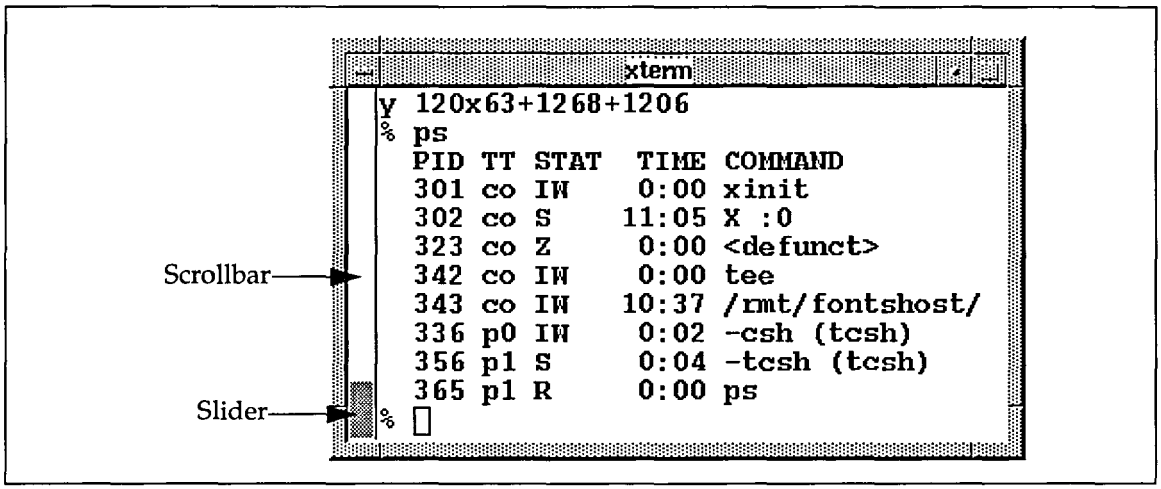
In addition to terminal emulation, xterm provides features such as an optional scrollbar, the ability to copy and paste with the mouse, and four menus.

Using the xterm scrollbar

The scrollbar enables you to review text that has scrolled up past the top of the window. To create an xterm window with a scrollbar, start an xterm from a command line with the `-sb` option: `xterm -sb &`

An xterm window with a scrollbar is shown in Figure 23.

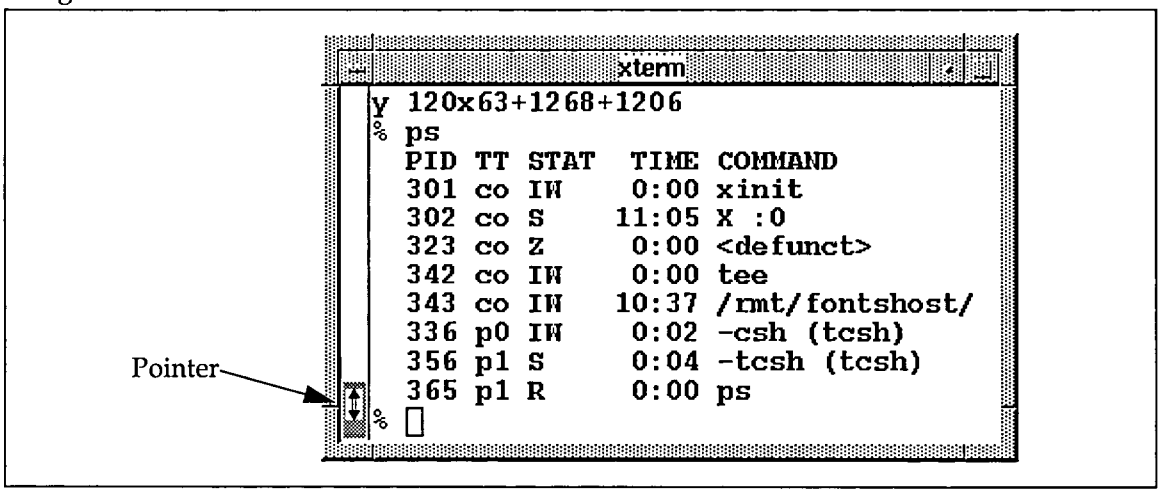
Figure 23
xterm scrollbar



To review text that has scrolled past the top of the window:

1. Place the mouse pointer over the scrollbar slider as shown in Figure 24. The pointer becomes an up and down arrow.
2. Hold down the middle mouse button and drag the slider with the mouse pointer up towards the top of the window until the needed text is showing.
3. Move the mouse pointer up and down until the text is in the desired place.
4. Release the mouse button.
5. You can scroll back to the command line with the mouse, or you can press SPACEBAR to return quickly to the command line.

Figure 24
Using the scrollbar



Before any text scrolls above the window, the slider is the full length of the window. As lines of text scroll past the top of the window, the slider shrinks to indicate that you can scroll backwards.

By default, xterm saves the last 64 lines that have scrolled above the window. You can change the number of scrolled lines that xterm saves by using the `-sl` command line option. For example:

```
xterm -sl 100
```

Using copy and paste

xterm enables you to use the mouse to copy sections of text from an xterm window to a buffer and then paste the text into another xterm window.

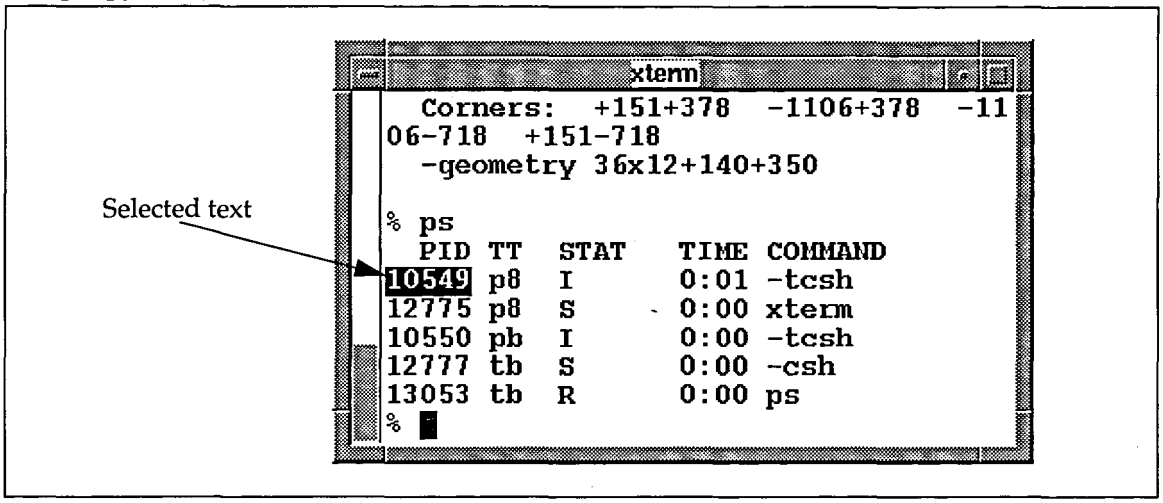
There are several ways to select text. To copy a section of text:

1. Move the mouse pointer to the beginning of the section you want to copy.
2. Press the left mouse button and drag the mouse pointer over the text that you want to copy.
3. Release the left mouse button when all the text you want to copy is highlighted in reverse video. When you release the mouse button, the text is placed in the copy buffer.

Another way to copy text is to:

1. Move the mouse pointer to the beginning of the section you want to copy.
2. Click the left mouse button.
3. Move the mouse pointer to the end of the section you want to copy.
4. Click the right mouse button. The section of text becomes highlighted in reverse video, as shown in Figure 25.

Figure 25
Using copy and paste



You can copy a single word by double-clicking the left mouse button on the word.

You can copy a single line by triple-clicking the left mouse button on the line.

You can only paste the last text that you highlighted. xterm replaces its copy buffer each time you select new text. To paste text that you have copied with the mouse pointer:

1. Place the mouse pointer where you want to insert your text and press the left mouse button to set the insertion point.
2. Press the middle mouse button. The text appears in the xterm window.

You can copy and paste to and from some other X clients as well, but copy and paste may not work the same way it does for xterm.

Using xterm menus

Four xterm menus enable you to invoke or change xterm features. The xterm menus include:

- **Main Options**—Enables you to set xterm modes and to send signals that affect the xterm process.
- **VT Options**—Enables you to choose setup options.
- **VT Fonts**—Enables you to change the xterm's font.
- **Tek Options**—Enables you to set options for the Tektronics window. This window is rarely used.

You can display the first three menus by placing the mouse pointer in the xterm window, pressing the CTRL key, and pressing the left, middle, or right mouse button respectively. You can display the Tek Options menu through an option on the VT Options menu.

To select an option from a menu:

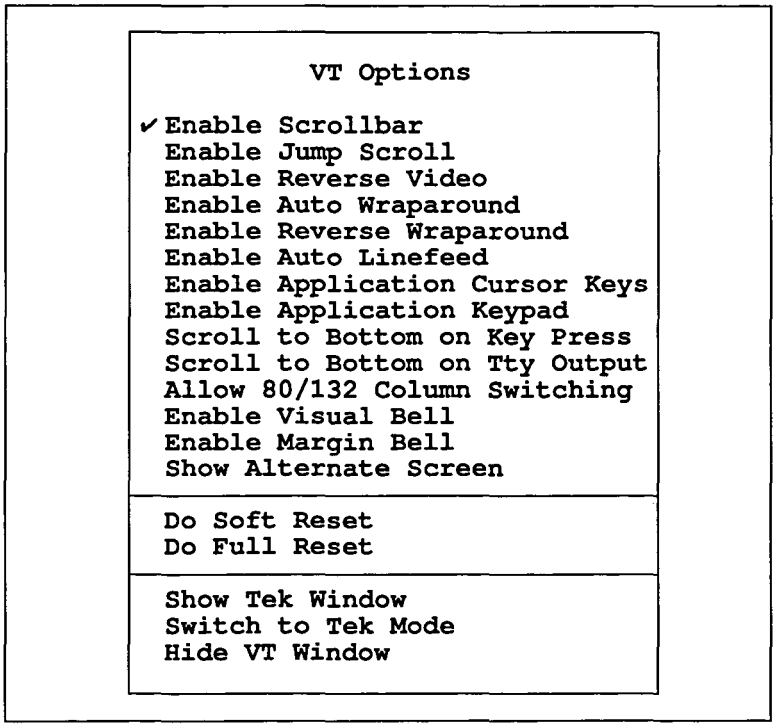
1. Continue to hold down the left mouse button after displaying the menu.
2. Drag the mouse down through the menu. Each option is highlighted in reverse video as the mouse pointer passes over it.
3. Release the mouse button while an option is highlighted to choose the option.

Use this same procedure to deselect an option.

You will seldom use the Main Options or Tek Options menu. For more information on these two menus and their options, refer to the man page for xterm.

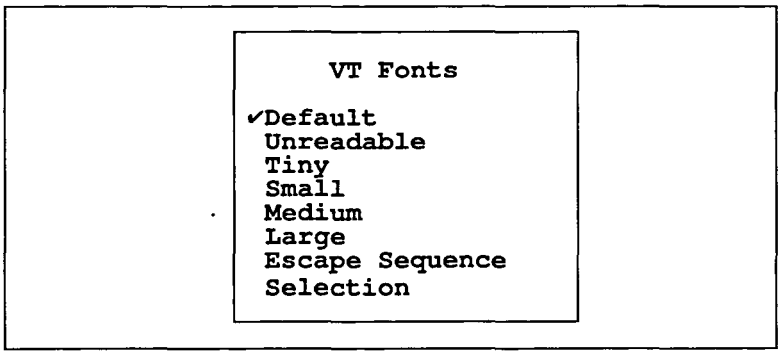
The VT Options menu is shown in Figure 26. The most useful options include enabling and disabling the scrollbar and reverse video. Enabled options are preceded by a check mark.

Figure 26
VT Options menu



The VT Fonts menu is shown in Figure 27. With this menu, you can change the font in your xterm window.

Figure 27
VT Fonts menu



Specifying X client defaults with resources

3

This chapter teaches you how to set your own default options that affect the appearance and behavior of X clients.

The "Overview" section of this chapter teaches you how to set simple X client default options. If you want to change a font, a color, or anything else that affects an X client as a whole, then you only need to read the "Overview" section, which includes "How resource specifications work," "The easy way to set X defaults," and "Examples of common resource specifications."

If you want to learn more about setting complex defaults that affect specific parts of an X client, then you should read the entire chapter.

Overview

Most X clients have many options that you can set. In the previous chapter, you read about using command line options to specify scrollbars, window size, fonts, reverse video, and so on. Most X clients have many more options, but it becomes tedious to specify these options on the command line. Instead, X lets you specify these options as defaults in a file in your home directory.

Each option of an X client is represented by a variable called a *resource*. By setting the value of a resource variable, you can change the appearance or functionality of an X client. For example, in the last chapter we discovered that the command `xterm -sb` creates an xterm window with a scroll bar. If you want xterm to always have a scrollbar without having to use the `-sb` option, you could place the following resource specification in a resource file in your home directory:

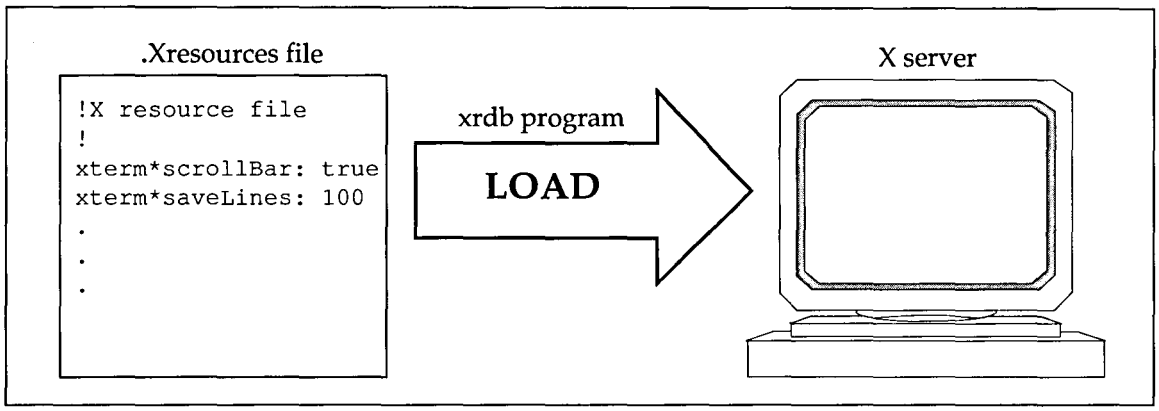
```
xterm*scrollBar: true
```

Once this resource is loaded into your X server's resource manager, all you have to do is invoke xterm, and an xterm window will appear with a scrollbar. The resource manager is the part of your X server that keeps a database of resource specifications.

How resource specifications work

First you place resource specifications in a resource file (usually called .Xresources or .Xdefaults[1]). Then you load these specifications into your X server's resource manager using a program called xrdb, the X resource database utility. This is usually done in your .xsession or .xinitrc file so that the resource specifications are loaded every time you start X. When you start a client, your X server looks in the resource manager's database to find resource specifications for that client. This process is shown in Figure 28.

Figure 28
How resources work



The easy way to set X defaults

This section teaches you how to write simple resource specifications.

First, decide which X client and options you want to change. To get a list of the resources that you can change, look at the X client's man page under the RESOURCES or X DEFAULTS heading.

For example, if you want to change the font that the xterm client uses, look at the man page for xterm under the title RESOURCES. There are many resources for the xterm client, but if you look closely, you will find a resource named `font`.

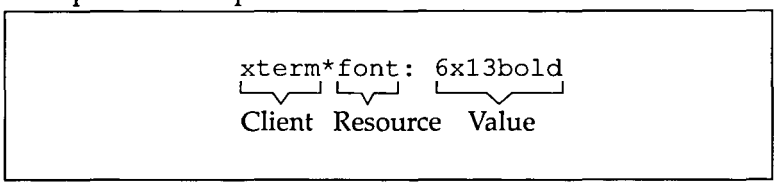
Once you decide on the client and option:

1. Edit your .Xresources file.
2. Starting in column 1, type in the X client's name followed by an asterisk (*), followed by the name of the resource that you

[1]You can choose any name for your X resource file; this book uses .Xresources.

want to set, followed by a colon. Then, skip at least one space or use a tab, followed by the value you want the resource variable to have as shown in Figure 29.

Figure 29
Example resource specification



Note

Your resource file must have a carriage return at the end of the last line. Otherwise xrdb will not load the last line.

3. Save your .Xresources file.
4. At the command line, enter:

   xrdb -load .Xresources

   from your home directory. The next time you start this client, it will have the specified characteristic.
5. Look at your X start-up file. If it does not contain a statement similar to the one below, then you should add it.

   ```
   xrdb -load $HOME/.Xresources
   ```

   For more information on X initialization files, refer to Chapter 6, " Writing an X start-up script."

Examples of common resource specifications

This section contains a sample resource file with some simple resource settings.

```
!An exclamation point(!) denotes a comment line.
!The following line enables the scrollbar on any
!X client that can have a scrollBar.
*scrollBar: true

!Settings for xterm
xterm*font: 6x13bold
xterm*saveLines: 100

!Enable visual bell for xpostage
xpostage*visualBell: true

!Set xclock colors

xclock*background: blue
xclock*foreground: white
```

Writing detailed resource specifications

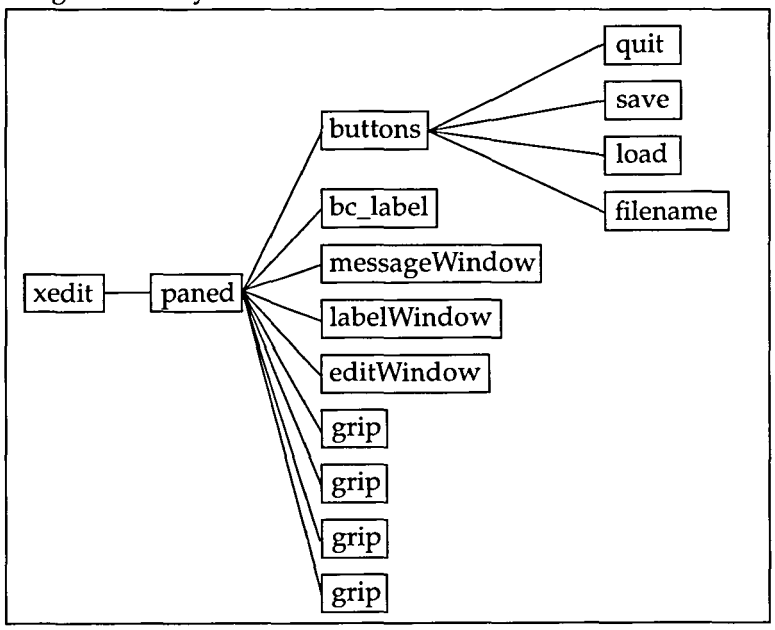
The rest of this chapter gives you the information needed to write detailed resource specifications.

What is a widget?

Before setting more complex resource specifications, you should know about widgets and widget hierarchies. Windows are made up of widgets. For example, menus, buttons, and text areas are widgets. When programmers build windows, they build many widgets on top of each other. This creates a widget hierarchy. Knowing a widget's hierarchy is useful when you want to set resource variables.

Figure 30 shows the widget hierarchy for the xedit client. Each box represents a widget. Notice that the widget named quit is built on top of the widget named buttons. The widget named buttons is built on top of the widget named paned, and so on. Therefore the hierarchy for the quit widget is xedit.paned.buttons.quit. The dots (.) indicate the next level in the hierarchy. Widget names separated by a dot must be next to each other in the widget hierarchy.

Figure 30
Widget hierarchy for the xedit client



What are widget classes?

All widgets belong to a *class*. A widget class is simply a group of widgets that have similar characteristics. The widgets in a class are called *instances* of that class. For example, the quit, save, and load widgets in xedit are instances in the class named Command, as shown in Table 1.

Class names always begin with a capital letter. In some cases, a widget's instance name is the same as its class name, except for the initial capital letter. For example, the paned widget is an instance in the class Paned.

Table 1
Widget classes and instances in xedit

Classes	Xedit	Paned	Label	Text	Command
Instances	xedit	paned	bc_label	messageWindow	quit
		buttons	labelWindow	editWindow	save
				filename	load

When you refer to a widget class in a resource specification, you are referring to all of the widgets in that class. For example, when you refer to the class Command in xedit, you are referring to the quit, save, and load widgets.

Using xedit's hierarchy shown in Figure 30, the following specification makes xedit's quit button appear in the 9x15 font:

```
xedit.paned.buttons.quit.font: 9x15
```

If you replace quit in the previous specification with its class name Command, then the specification changes the font of every widget in the class Command that is also next to the buttons widget in the widget hierarchy:

```
xedit.paned.buttons.Command.font: 9x15
```

This resource specification makes the quit, save, and load buttons appear in the 9x15 font.

Refer to a client's man page to:

- Learn a client's widget hierarchy
- Learn widget class and instance names

Syntax for setting resources

The syntax for setting a resource is:

Xclient.*widget_hierarchy*.*attribute*: *value*

where

Xclient Is the name of an X client. If you omit the X client name, the resource variable applies to all clients that have the specified attribute and the same hierarchy.

widget_hierarchy Is the widget hierarchy leading to the widget whose attributes you want to change.

attribute Is the name of the resource variable such as font, scrollbar, or background. The attribute must be followed by a colon.

value Is the value that you are giving the resource variable such as a font name, a color, true, false, and so on.

Entries that begin with ! are considered comments and are ignored.

Note

The resource manager provides few or no error messages. If you specify a resource incorrectly, the resource will not take effect.

Using tight and loose bindings

The term *binding* refers to the way that you link parts of a resource specification together to specify a widget hierarchy. The parts of a resource specification include widget instance names and/or class names. You can bind these parts of a resource specification in two ways:

- Tight bindings are very specific and are represented by a dot (.).
- Loose bindings are less specific and are represented by an asterisk (*).

If you use a tight binding in a resource specification, names on either side of the period must be next to one another in the widget hierarchy, or the resource specification will be ignored. An example of a tight binding is

```
xedit.paned.buttons.quit.font: fixed
```

The previous specification makes the quit button of the xedit client appear in the font named fixed.

If you use a loose binding in a resource specification, you can omit any number of levels of the widget hierarchy between two names that are separated by an asterisk. An example of a loose binding is

```
xedit*quit*font: fixed
```

The previous specification makes *any* xedit client button named quit appear in the font named fixed.

The two previous specifications have the same effect; however, the second entry was easier to enter because we did not have to know the entire widget hierarchy. To find out what an X client's widget hierarchy is, refer to the client's man page or use the editres client as discussed in "Using editres" on page 44.

Loose bindings are better than tight bindings

In most cases, loose bindings are better than tight bindings because:

- Loose bindings are easier to write than tight bindings because you do not need to know as much of the X client's widget hierarchy.
- A client's widget hierarchy may change from release to release. If the hierarchy changes, your tight bindings may no longer be correct. For example, in the current version of xedit, the following specification is valid:

  ```
  xedit.paned.buttons.filename.font: 6x13bold
  ```

  However, in a future version of xedit, its widget hierarchy might change so that the file name widget is built on top of the paned widget rather than the buttons widget. Thus, the previous specification would be invalid. The following specification would work in either case:

  ```
  xedit*filename*font: 6x13bold
  ```

  Of course if the widget hierarchy changes radically, even the previous specification may not work.

Combining tight and loose bindings

You can use tight and loose bindings in the same specification. For example:

```
xedit*filename.font: fixed
```

You can also begin a specification with an asterisk (*) to make it apply to all clients with the specified attribute.

Note

In most cases, you should avoid creating resource specifications that begin with an asterisk (*) unless you actually want the specification to apply to any and all X clients.

Precedence rules

The following rules enable you to make resource specifications for both specific and general cases:

1. Tight bindings are more specific than, and take precedence over, loose bindings.
2. Instance names are more specific than, and take precedence over, class names.
3. More specific specifications take precedence over less specific specifications.

Consider the following:

```
xedit.paned.buttons.quit.font: 9x15bold
xedit.paned.buttons*font: fixed
```

At first glance, these two specifications seem to conflict. The first line makes the quit button appear in the font named 9x15bold. The second line makes all of the buttons in that hierarchy have the font named fixed. The first line takes precedence over the second line because the first line is more specific. So, all buttons will have the font named fixed, except for the quit button, which will have the font named 9x15bold.

If the previous two specifications were placed into the resource file in reverse order, they would still have the same effect.

Using client names in resource specifications

Most resource specifications begin with the name of the client. The name of the client is usually the name that you use to start the client. For example, because you start the xedit client with `xedit &`, you also begin xedit resource specifications with the name xedit:

```
xedit*quit*font: fixed
```

With the `-name` command line option, you can start most clients with a different name. For example,

```
xedit -name doc &
```

starts the xedit client with the name doc. This means that resource specifications that begin with doc rather than xedit will apply to this client. However, any resource specification that begins with the class name Xedit will apply to the xedit client regardless of the name it starts with.

Examples of detailed resource specifications

Consider the resource specifications in Figure 31 for the xedit client. Lines that start with an uppercase X pertain to widgets in the Xedit class. Because the class name is specified, these resource specifications apply to all xedit clients, regardless of the names used to start them.

Line 7 in Figure 31 only applies to X clients that are named qedit. Therefore, line 7 only applies to xedit when it is started with `xedit -name qedit &`. It is often useful to specify resources for X clients with custom names. Each custom name can invoke different client characteristics for a specific purpose.

Figure 31
xedit client resource specifications

```
1 Xedit*Commands*font: 9x15bold
2 Xedit*quit*background: red
3 Xedit*save*background: green
4 Xedit*load*background: blue
5 Xedit.paned.bc_label.font: -adobe-helvetica-bold-r-normal--34-240-100-100-..
6 Xedit.paned.labelWindow.font: -sony-fixed-medium-r-normal--24-230-75-75-c-..
7 qedit.paned.buttons.quit.background: black
```

Looking at Figure 31[1]:

- Line 1 makes all buttons in xedit appear in the 9x15bold font.
- Line 2 causes any widget named quit to have a red background.
- Line 3 causes any widget named save to have a green background.
- Line 4 causes any widget named load to have a blue background.
- Line 5 causes the widget named bc_label to have the specified bold font.
- Line 6 causes the widget named labelWindow to have the specified medium font.
- Line 7 causes the button named quit to have a black background if the X client is named qedit. A better way to write line 7 would be to use loose bindings:

  ```
  qedit*quit*background: black
  ```

[1]The line numbers in this figure are for reference purposes and do not normally appear

Setting resources from the command line

Before you place a resource specification in your resource file, you can test the specification at the command line with the `-xrm` option when you start the client. For example, you can invoke xedit with

```
xedit -xrm 'xedit*quit.font: 9x15bold'
```

to see if this specification works before you place it in your resource file. Resources that you set in this way are not added to the resource manager's database. The resource only affects the current instance of the client.

When you specify a resource with the `-xrm` option, the resource will not take effect if a resource that takes precedence over it has already been loaded into the resource manager. For example:

```
xterm -xrm 'XTerm*font: 9x15bold' &
```

will not take effect if the following specification is in your resource file:

```
xterm*font: fixed
```

This is because the instance name in the resource file (`xterm*font`) takes precedence over the class name (`Xterm*font`) specified on the command line.

However, you can always override resource specifications by specifying any other command line options. For example:

```
xterm -fn 9x15bold &
```

will always work regardless of what is in your resource manager's database.

Using `xrdb` to set resources

This section teaches you how to use the xrdb utility to load resource specifications into your X server's resource manager.

The syntax for the `xrdb` command is

`xrdb [`*option*`]`

The xrdb program has many options, but the most practical ones are presented here. For a list of all options, refer to the xrdb man page or enter `xrdb -help`.

Loading resources with the `-load` option

Normally, you will find a line similar to

```
xrdb -load $HOME/.Xresources
```

in your .xsession or .xinitrc X start-up file. The `-load` option deletes all of the resource specifications in the resource manager's database and replaces them with the specifications in the specified file, usually your .Xresources file.

You can also use `xrdb -load` *filename* from the command line to reload your resource specifications after you make a change in your resource file.

Note

Your resource file must have a carriage return at the end of the last line. Otherwise, xrdb will not load the last line.

Listing active resources with the `-query` option

You can find out what your current resource specifications are by using the `-query` option as shown in Figure 32. The `-query` option lists the resource settings from the resource manager's database rather than what is in your .Xresources file.

The resource manager accepts meaningless resource specifications as long as they use the correct syntax. Having a specification in the resource manager's database does not necessarily mean that the specification is valid.

Figure 32
Example using the `-query` option

```
% xrdb -query
*highlightColor: XtDefaultBackground
*load.update: 61
Cxdb.commandButtons: true
Cxdb.commandMenu: true
XPostage*visualBell: true
Xclock*geometry: 120x63+1311+1201
xterm*font: -adobe-courier-bold-r-normal--14-140-75-75-m-90-iso8859-1
xterm*scrollBar: on
```

To list resource specifications for a single client, use the `appres` command as explained on the next page.

Adding new resources with the `-merge` option

The `-merge` option enables you to add resources to the resource manager's database without removing the ones that are already there. For example, your .Xresources file has already been loaded by xrdb in your .xsession or .xinitrc file. You have put two new resource specifications in a file called newres. To add the resource specifications in newres to the ones already in your resource manager's database, you would enter:

```
xrdb -merge newres
```

If a new specification matches one that is already in your resource manager's database, the `-merge` option replaces the old specification with the new one. For example, assume that

```
xterm*font: 9x15bold
```

is currently in your resource manager's database. If you use the `-merge` option with a file that contains

```
xterm*font: fixed
```

then fixed replaces 9x15bold in your resource manager's database.

Listing `xrdb` options with the `-help` option

The `-help` option displays a list of available options that you can specify with the `xrdb` command.

Listing a single client's resources with `appres`

The `appres` command is often used with xrdb. The `appres` command lists all of the resources in the resource manager's database that apply to a particular client. For example, to find out what resources you have set for xterm, enter `appres xterm` as shown in Figure 33.

Figure 33
Example of the `appres` command

```
% appres xterm
xterm*font: -adobe-courier-bold-r-normal--14-140-75-75-m-90-iso8859-1
xterm*saveLines: 100
xterm*scrollBar: on
*highlightColor: XtDefaultBackground
```

Using editres

The editres client is a dynamic resource editor for X clients. This tool enables you to view the widget hierarchy of any X client that speaks the editres protocol. This includes all of the clients in X11 Release 5; however, you do *not* need a Release 5 version of the X server to use editres.

The editres client helps you build resource specifications and enables you to apply the resource to an X client and view the results. When you finish building a resource specification, editres can save or append the resource string to your .Xresources or any other file.

In this section:

- **Getting started**—Tells you about the features in editres.
- **Sample session using editres with xedit**—Shows you a practical approach to using editres.

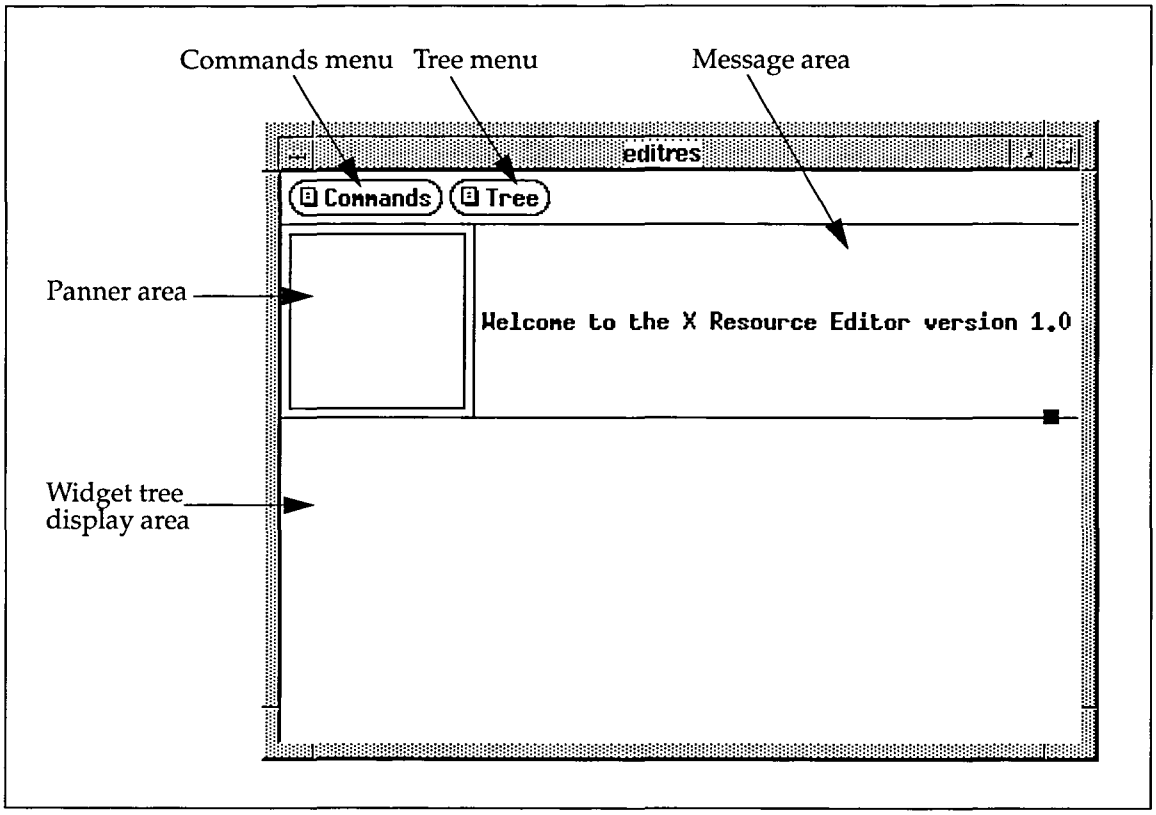
Getting started

Start the client you wish to use editres on, then enter

```
editres &
```

at your command line. The window in Figure 34 appears.

Figure 34
editres window



The editres window has the following features:

- **Commands menu**—Displays editres command options.
- **Tree menu**—Displays options that affect the widget tree.
- **Message area**—Displays messages after you issue commands.
- **Panner area**—Enables you to pan through sections of a large widget tree with the mouse. A panner is a two-dimensional scrollbar.
- **Widget tree display area**—Displays the widget tree of a selected client.

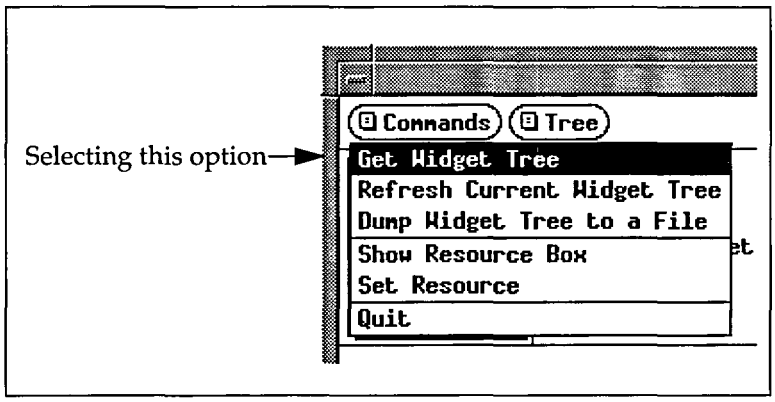
This section uses the xedit client as an example. However, editres works with any R5 toolkit-based X client.

Using the menus

To display one of the menus and select a menu item:

1. Move the mouse pointer over one of the menu titles.
2. Press and hold down the left mouse button. A menu similar to the one in Figure 35 appears.
3. While holding down the left mouse button, move the mouse pointer over the option you want. The option is highlighted.
4. Release the mouse button while the option is highlighted. editres performs the command.

Figure 35
Commands menu



Displaying an X client's widget tree

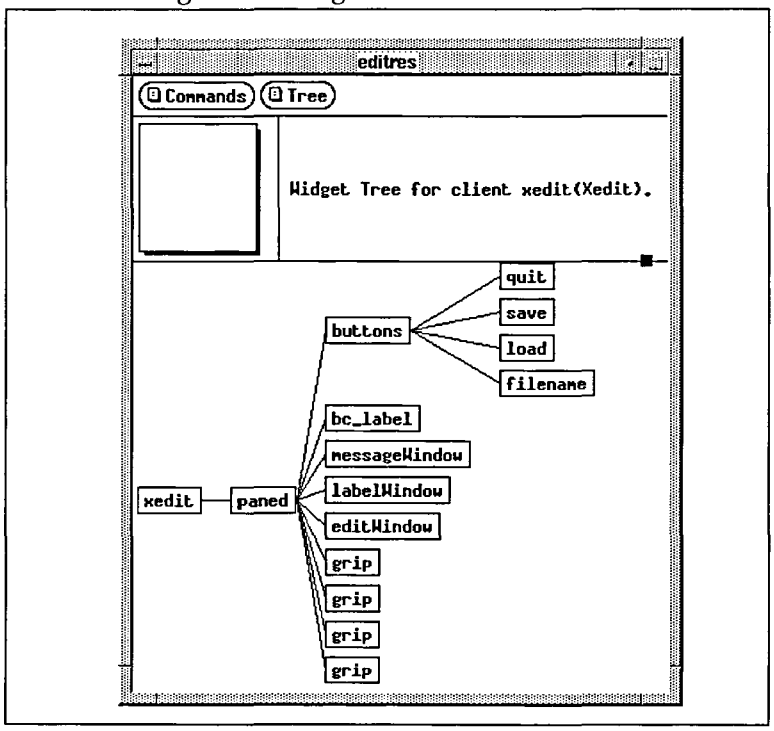
Before you can specify complex resource specifications for an X client, you often need to know the client's widget hierarchy. With editres, you can display a client's widget hierarchy by choosing the Get Widget Tree option.

To display a client's widget tree:

1. Select the Get Widget Tree option from the Commands menu as shown in Figure 35. The mouse pointer becomes a crosshair.
2. Move the pointer into the window of a client and click the left mouse button. If the client understands the editres protocol, the client's widget tree will appear in the widget tree display area as shown in Figure 36. If the client does not understand the editres protocol, you get the following message after it times out; this may take a while:

```
It appears that this client does not
understand the editres protocol.
```

Figure 36
editres showing xedit's widget tree



This widget tree shows the instance names for each widget in the client. To show the class names, select the Show Class Names option from the Tree menu.

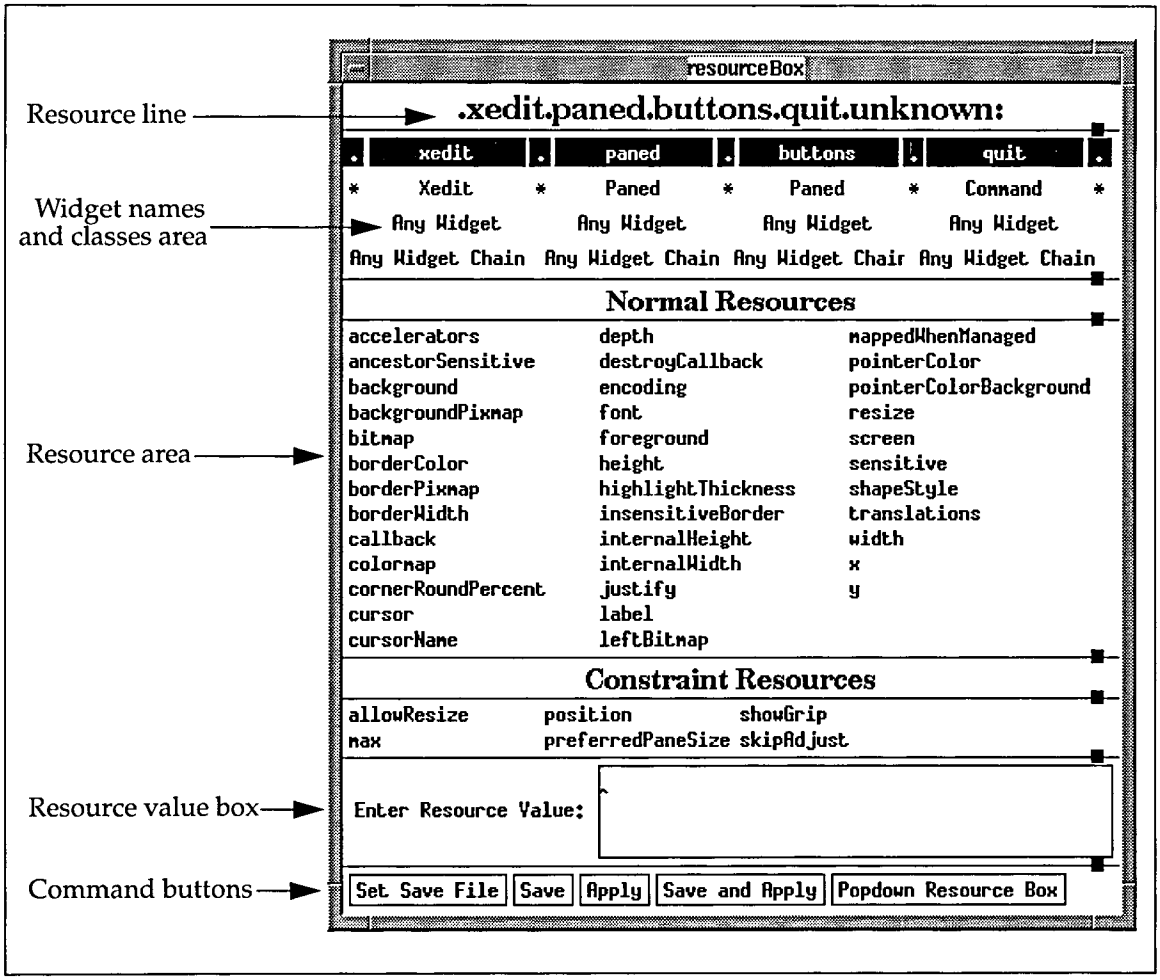
Displaying the resource box

The resource box displays all of the resources available for a selected widget. It also enables you to change resources dynamically and save them.

To display the resource box:

1. Select a widget in the tree by moving the mouse pointer over a widget and pressing the left mouse button. The widget is highlighted in reverse video. Only one widget should be highlighted.
2. Select the Show Resource Box option from the Commands menu. The resource box appears as shown in Figure 37.

Figure 37
Resource box



The resource box has the following features:

- **Resource line**—Shows the resource specification that you have built using the other features of the resource box.
- **Widget names and classes area**—Displays the possible specification combinations that you can choose for a widget. By clicking the left mouse button on cells in this area, you can change the resource specification displayed in the resource line.
- **Resource area**—Displays the resources that this widget has. You can choose a resource by clicking on one with the left mouse button. The resource becomes highlighted.

- **Resource value box**—Enables you to enter a value for the resource you have chosen in this box. Do not press RETURN in this box.
- **Command buttons**—Enables you to apply and save resource specifications as well as close the resource box.

Sample session using editres with xedit

This sample session uses editres to set a resource that will change the font of the widget in the xedit client that is shown in Figure 38.

1. Display the widget tree for xedit by choosing the Get Widget Tree option from the Commands menu. The mouse pointer becomes a cross.
2. Move the mouse pointer into the xedit window.
3. Click the left mouse button. The xedit widget tree appears in the editres window.
4. Choose the Select Widget in Client option from the Tree menu as shown in Figure 39. The mouse pointer becomes a cross.

Figure 38
The xedit client

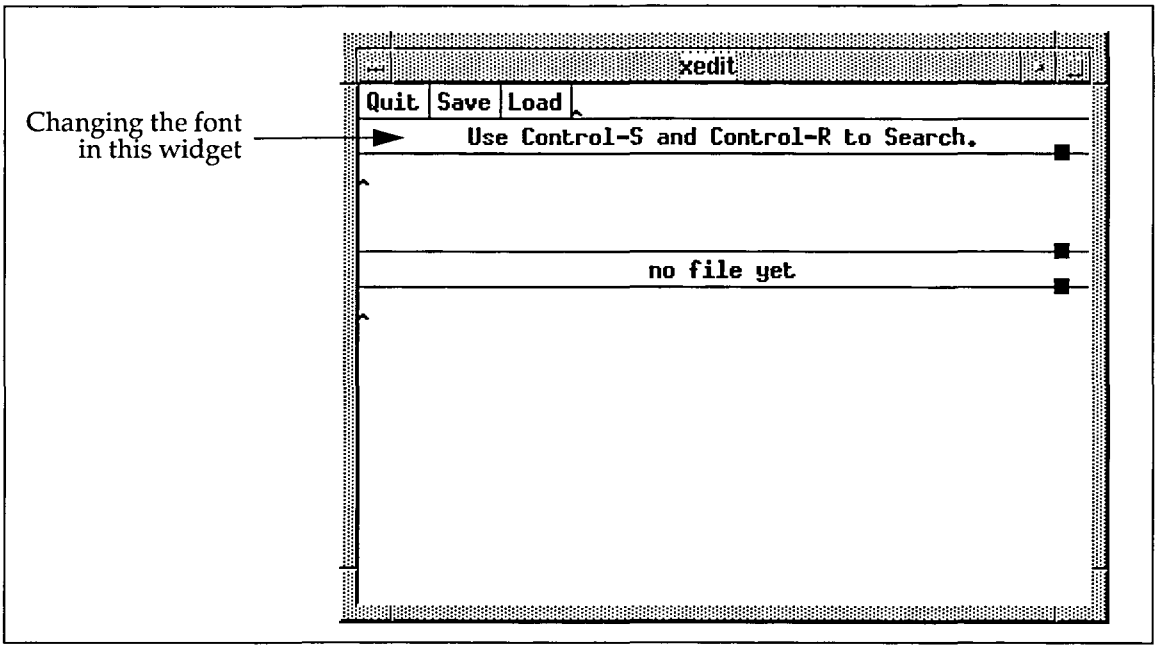
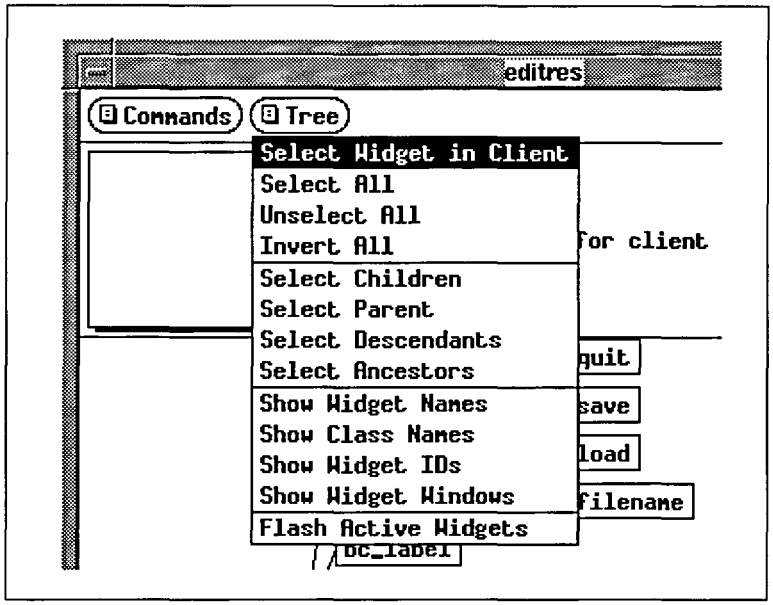


Figure 39
Tree menu

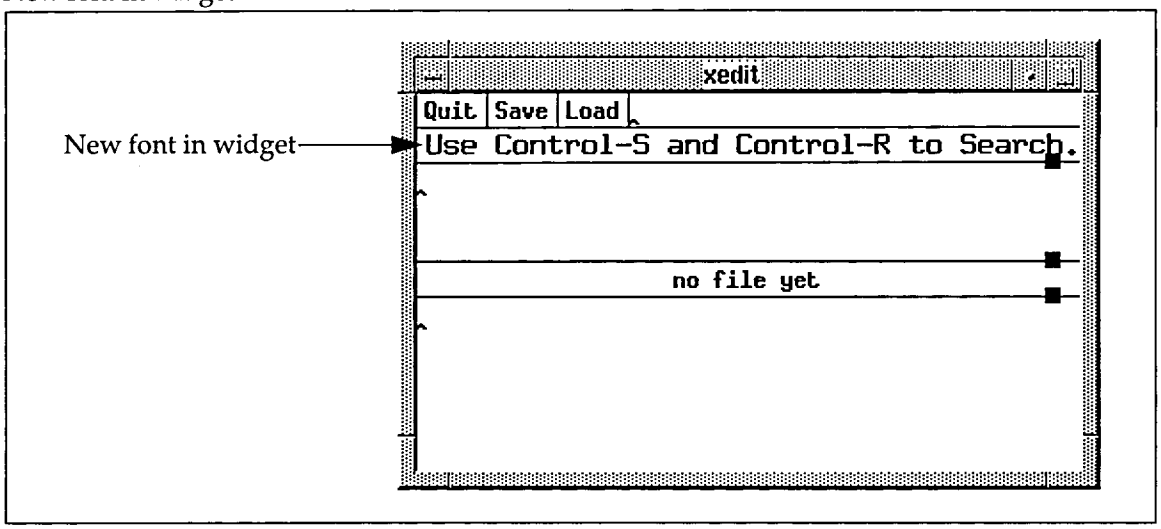


5. Move the mouse pointer into the xedit client over the area containing the text "Use Control-S and Control-R to Search."
6. Click the left mouse button. Notice that the widget begins to flash. The matching widget in the editres widget tree (bc_label) is highlighted for you.
7. Choose the Show Resource Box option from the Commands menu. The resource box appears for the bc_label widget.
8. In the Normal Resources section of the resource box, click the font resource. The font resource is highlighted.
9. In the Enter Resource Value box at the bottom of the resource box, type

   ```
   9x15bold
   ```

   or another font name, but do not press RETURN.
10. Press the Apply button at the bottom of the resource box. In the xedit window, notice that the font changes in the specified widget as shown in Figure 40.

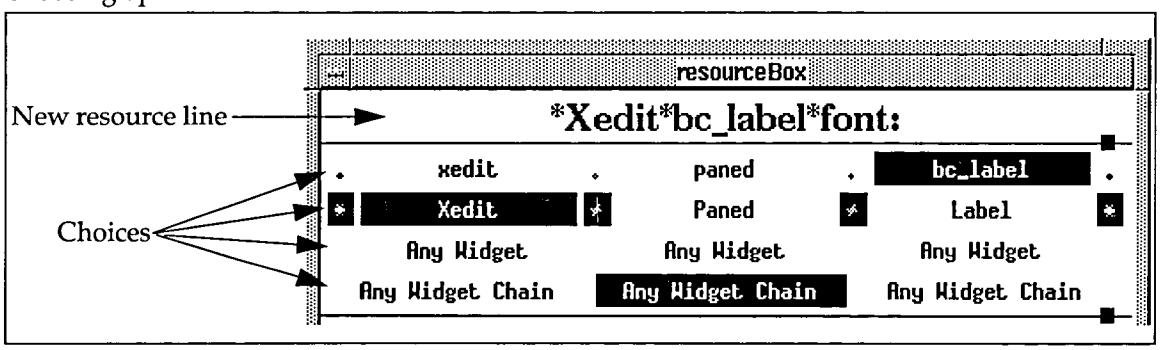
Figure 40
New font in widget



11. In the resource names and classes area at the top of the resource box, click each asterisk to make this a loosely bound specification as shown in Figure 41.
12. In the same area, click the Xedit class name to make this specification apply to any xedit client.
13. Click Any Widget Chain under paned. Notice that the resource line now shows

```
*Xedit*bc_label*font:
```

Figure 41
Choosing options



14. Press the Apply button again to make sure that this resource specification is correct. Notice that the editres message window displays the message:

```
SetValues was Successful.
```

If you make a mistake, you get an error message.

15. Press the Save button at the bottom of the resource box. The file dialog box appears.
16. Enter the file name where you want to save resource specifications and press okay. If you enter an existing file, it will append your resource specifications to this file.
17. Close the resource box by pressing the Popdown Resource Box button.
18. Quit editres by choosing the Quit option from the Commands menu.

Customizing the mwm window manager

4

This chapter teaches you how to:

- Modify and create mwm menus
- Set mouse button functions
- Set function key combination bindings
- Set mwm resources

To modify mwm, you make changes to two files:

- **.mwmrc**—Contains mwm menu specifications, mouse button bindings, and key bindings.
- **.Xresources**—Contains resource specifications for individual clients and for mwm.

After you make changes to these files, you can activate the changes by reloading your .Xresources file with the following `xrdb` command:

```
xrdb -load $HOME/.Xresources
```

and then restarting mwm with the Restart... option from the Root Menu. You can also restart mwm by logging out and logging back in.

Modifying your .mwmrc file

If you do not have a .mwmrc file in your home directory, then copy the file /usr/lib/X11/system.mwmrc to your home directory, and rename it .mwmrc. Figure 42 shows the default system.mwmrc file. This file contains three sections:

- **Menu specifications**—Define the content of the Root Menu and the window menu.
- **Key bindings**—Map keystrokes to window manager functions.
- **Button bindings**—Map mouse button presses to window manager functions.

Figure 42
Default system.mwmrc file

```
## DEFAULT mwm RESOURCE DESCRIPTION FILE (system.mwmrc and .mwmrc)
#
# menu pane descriptions
#
# The Root Menu description defines the content of the Root Menu.
Menu RootMenu
{
 "Root Menu"            f.title
 "New Window"           f.exec "xterm &"
 "Shuffle Up"           f.circle_up
 "Shuffle Down"         f.circle_down
 "Refresh"              f.refresh
 no-label               f.separator
 "Restart..."           f.restart
 "Exit"                 f.quit_mwm
}

# The Default Window Menu description defines the
# content of the default window menu.
Menu DefaultWindowMenu
{
 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>F9    f.minimize
 Maximize       _x      Alt<Key>F10   f.maximize
 Lower          _L      Alt<Key>F3    f.lower
 no-label                             f.separator
 Close          _C      Alt<Key>F4    f.kill
}

#
# The Key Binding descriptions map keystrokes to window manager functions.
#
Keys DefaultKeyBindings
{
 Shift<Key>Escape           window|icon          f.post_wmenu
 Meta<Key>space             window|icon          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 Shift Ctrl<Key>exclam root|icon|window     f.set_behavior
 Meta<Key>F6                window               f.next_key transient
 Meta Shift<Key>F6          window               f.prev_key transient
 <Key>F4                    icon                 f.post_wmenu
}
```

Figure 42 (continued)
system.mwmrc file

```
# Button binding descriptions map button presses to window manager functions.
#
# The default button bindings section
Buttons DefaultButtonBindings
{
 <Btn1Down>     icon|frame f.raise
 <Btn3Down>     icon       f.post_wmenu
 <Btn1Down>     root       f.menu RootMenu
}

# This section is an alternate set of button bindings that
# are not active unless you specify them in your resource file.
Buttons ExplicitButtonBindings
{
 <Btn1Down>          frame|icon     f.raise
 <Btn3Down>          frame|icon     f.post_wmenu
 <Btn1Down>          root           f.menu RootMenu
 Meta<Btn1Down>      window|icon    f.lower
!       Meta<Btn2Down>window|icon  f.resize
!       Meta<Btn3Down> window|icon f.move

}

# This section is an alternate set of button bindings that
# are not active unless you specify them in your resource file.
Buttons PointerButtonBindings
{
 <Btn1Down>          frame|icon     f.raise
 <Btn3Down>          frame|icon     f.post_wmenu
 <Btn1Down>          root           f.menu RootMenu
 <Btn1Down>          window         f.raise
 Meta<Btn1Down>      window|icon    f.lower
!       Meta<Btn2Down> window|icon f.resize
!       Meta<Btn3Down>window|icon  f.move
}

#
# END OF mwm RESOURCE DESCRIPTION FILE
```

Modifying the Root Menu

The Root Menu appears when you hold down the left mouse button while the mouse pointer is over the root window. This menu's options affect any or all windows on your display.

The Root Menu's specification in your .mwmrc file has the following syntax:

```
# comment
Menu menuname
{
     "label" function
     "label" function
  .
  .
  .
}
```

where

#	Indicates a comment line.
Menu *menuname*	Specifies the name of the menu, usually RootMenu.
{	Signals the beginning of the menu content.
"label"	Specifies the text for a menu item that appears when you display the menu.
function	Specifies one of the window manager functions.
}	Closes the menu specification

Figure 43 shows the default Root Menu specification.

Figure 43
Default Root Menu specification

```
# Root Menu Description
Menu RootMenu
{
       "Root Menu"     f.title
       "New Window"    f.exec "xterm &"
       "Shuffle Up"    f.circle_up
       "Shuffle Down"  f.circle_down
       "Refresh"       f.refresh
       no-label        f.separator
       "Restart..."    f.restart
       "Exit"          f.quit_mwm
}
```

The mwm functions used in the Root Menu include:

- `f.title`—Specifies the title of the menu.
- `f.exec`—Executes the system command that follows it.
- `f.circle_up`—Raises the window on bottom to the top.
- `f.circle_down`—Lowers the window on top to the bottom.
- `f.refresh`—Redraws your entire display.
- `f.separator`—Creates a divider line in the menu.
- `f.restart`—Restarts mwm.
- `f.quit_mwm`—Stops mwm.

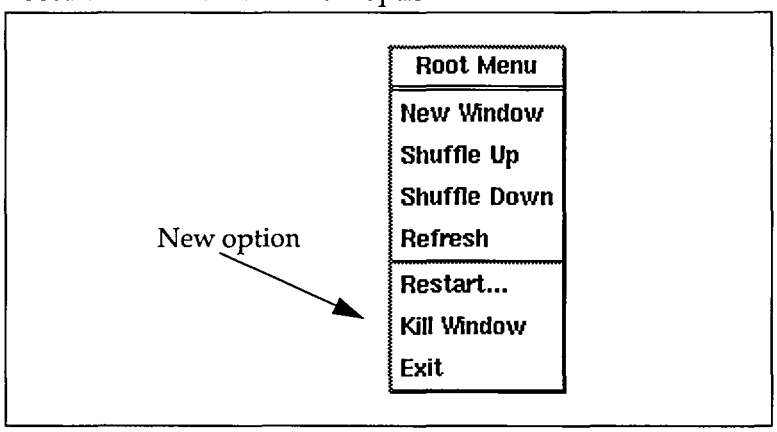
For a comprehensive list of mwm functions available for use in menus, refer to the mwm man page.

As an example, if you want to add a menu option named Kill Window that will allow you to kill windows with the mouse, add the following line:

```
"Kill Window"    f.exec "xkill" &
```

Before this new option shows up on the Root Menu, you must select the Restart... option from the Root Menu. After adding this option, the Root Menu will be similar to the one in Figure 44.

Figure 44
Root Menu with Kill Window option



Creating new menus

The Root Menu appears when you press the left mouse button while the pointer is over the root window. You can also create menus that appear when you press the right or middle mouse buttons over the root window.

To create a new menu, you must:

- Create a new menu specification
- Tell mwm when to display the menu with a resource

As an example, the specification in Figure 45 creates a menu that displays a few X utilities when you press the right mouse button while the mouse pointer is over the root window. This new menu is called Utilities. This menu uses the same syntax as the Root Menu.

Figure 45
New menu example

```
Menu Utilities
{
     "Utilities"      f.title
     "xlogo"          !"xlogo &"
     "xclip"          !"xclipboard &"
     "xclock"         !"xclock &"
     "xbiff"          !"xbiff &"
     "xcalc"          !"xcalc &"
     "xload"          !"xload &"
     "Backgrounds"    f.menu backgrnd
}
```

This menu contains two new things:

- The exclamation point (!) is a substitute for `f.exec`.
- The `f.menu` command instructs mwm to display another menu called backgrnd with a title of "Backgrounds." A submenu of this type is called a *cascading* menu. The specification for this menu is shown in Figure 46.

Figure 46
Backgrounds menu

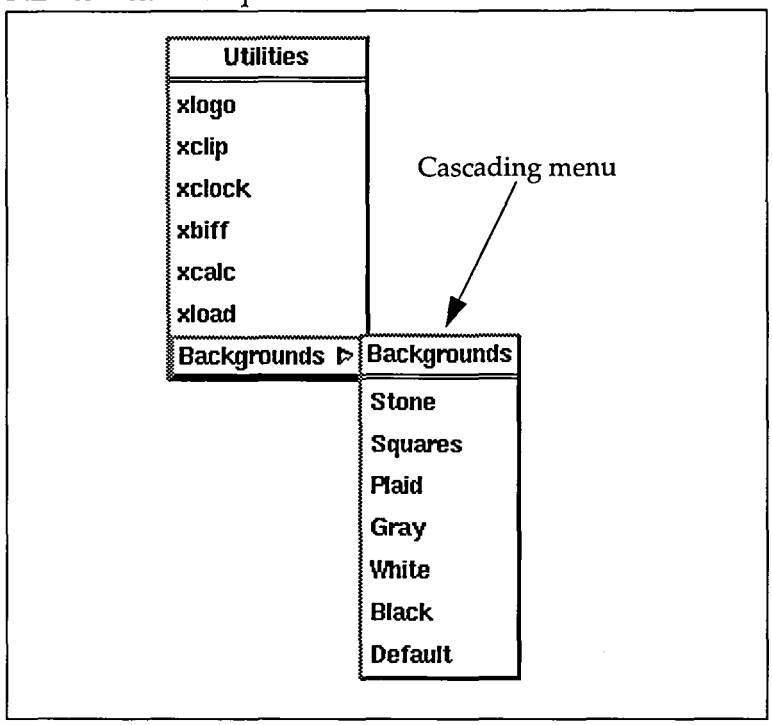
```
Menu backgrnd
{
     "Backgrounds"    f.title
     "Stone"          !"xgranite &"
     "Squares"        !"xsetroot -mod 16 16 -rv &"
     "Plaid"          !"xsetroot -mod 5 5 &"
     "Gray"           !"xsetroot -gray &"
     "White"          !"xsetroot -solid white &"
     "Black"          !"xsetroot -solid black &"
     "Default"        !"xsetroot &"
}
```

Now you need to tell mwm when to display the Utilities menu by placing the following line in the `Buttons` section of your .mwmrc file:

```
<Btn3Down>    root    f.menu Utilities
```

This line tells mwm to display the Utilities menu when you press the right (or third) mouse button. This menu would look like the one shown in Figure 47.

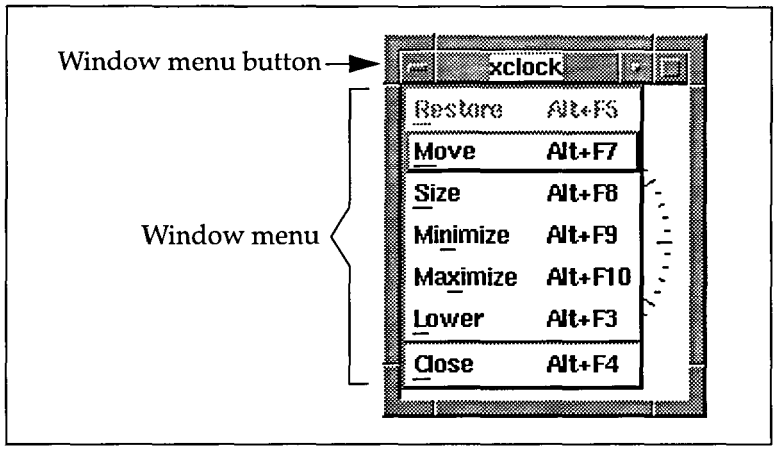
Figure 47
Utilities menu example



Modifying the window menu

Each window has a window menu that displays a list of commands that you can perform on that window. You display the window menu by pressing the window menu button with the left mouse button as shown in Figure 48.

Figure 48
Window menu



The window menu specification in your .mwmrc file has the following syntax:

```
#comment
Menu menuname
{
    "label"  _mnemonic  accelerator  function
    "label"  _mnemonic  accelerator  function
.
.
.
}
```

where

#	Indicates a comment line.
`Menu` *menuname*	Specifies the name of the menu.
{	Signals the beginning of the menu content.
"label"	Specifies the text for a menu item that appears when you display the menu.
_mnemonic	Specifies a unique letter abbreviation for the menu item label. The letter must appear in *label*, and you must precede the mnemonic with an underscore. Pressing the mnemonic while the menu is showing invokes the function.

accelerator	Specifies a key sequence that invokes the function without displaying the menu. For example, `Alt<Key>F9` is the accelerator that iconifies the window menu in Figure 49.
function	Specifies one of the window manager functions.
}	Closes the menu specification.

Figure 49 shows the default window menu specification.

Figure 49
Default window menu specification

```
# Default Window Menu Description

Menu DefaultWindowMenu
{
    "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>F9  f.minimize
    "Maximize"_x  Alt<Key>F10 f.maximize
    "Lower"   _L  Alt<Key>F3  f.lower
    "no-label"                f.separator
    "Close"   _C  Alt<Key>F4  f.kill
}
```

As an example, the following line changes the Close specification at the end of Figure 49:

```
Quit   _Q    <Key>F4    f.kill
```

This line makes Quit rather than Close appear on the menu. Q is now the mnemonic and the F4 function key is the accelerator.

The mwm functions used in the window menu include:

- `f.normalize`—Restores an iconified or maximized window to its normal size.
- `f.move`—Allows you to move a window with the mouse.
- `f.resize`—Allows you to resize a window with the mouse.
- `f.minimize`—Iconifies a window.
- `f.maximize`—Makes the window the size of your screen.
- `f.lower`—Lowers a window to the bottom of a stack of windows.
- `f.separator`—Creates a divider line in a menu.
- `f.kill`—Kills a client.

Creating alternate window menus

You can also create window menus that appear for specific clients. To create a window menu for a client called Xclient:

1. Create a menu with a unique menu name, for example, `XclientMenu`.
2. In your .Xresource file, enter:

```
Mwm*Xclient*windowMenu: XclientMenu
```

Figure 50 shows a customized menu for xterm.

Figure 50
xterm window menu

```
#custom xterm menu
Menu XtermMenu
{
      "Restore"    _e    <Key>F10    f.normalize
      "Iconify"    _I    <Key>F9     f.minimize
      "Raise"      _R    <Key>F8     f.raise
      "Lower"      _L    <Key>F7     f.lower
      "Move"       _M    <Key>F6     f.move
      "Size"       _S    <Key>F5     f.resize
      "Big"        _B    <Key>F4     f.maximize
      no-label                       f.separator
      "Quit"       _Q    Alt<Key>F3  f.kill
}
```

In the case of the menu in Figure 50, you must place the following resource specification in your .Xresources file:

```
Mwm*XTerm*windowMenu: XtermMenu
```

This resource specification tells mwm which menu to display when you display the window menu of an xterm window. To activate the line above and the menu in Figure 50, you must enter:

```
xrdb -load $HOME/.Xresources
```

Then, restart mwm with the Restart... option from the Root Menu.

Setting key bindings

In your .mwmrc file, you can also set keystroke combinations to window manager functions. For example, by default mwm displays the window menu when you hold down the **SHIFT** key and press the **ESC** key.

The key bindings section of your .mwmrc file has the following syntax:

```
#comment
Keys section_name
{
  [modifier_keys]<Key>keyname context function
  [modifier_keys]<Key>keyname context function
.
.
.
}
```

where

#	Indicates a comment line.
`Keys` *section_name*	Specifies the name of the key bindings section.
{	Signals the beginning of the section's content.
modifier_keys	Optional. Specifies that you must hold down either the **SHIFT** or **META** key while you type *keyname*.
keyname	Specifies the name of a key that will perform *function*.
context	Specifies where the mouse pointer can be for *function* to occur. The possible values are `window`, `icon`, and `root`. To specify more than one context, separate them with a vertical bar (|).
function	Specifies one of the window manager functions.
}	Closes the key binding specification.

Figure 51 shows the default key bindings section in the system.mwmrc file.

Figure 51
Default key bindings section

```
#
# key binding descriptions
#
Keys DefaultKeyBindings
{
  Shift<Key>Escape              window|icon        f.post_wmenu
  Meta<Key>space                window|icon        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 Shift Ctrl<Key>exclam    root|icon|window   f.set_behavior
  Meta<Key>F6                   window             f.next_key transient
  Meta Shift<Key>F6             window             f.prev_key transient
  <Key>F4                       icon               f.post_wmenu
}
```

As an example, the following line changes the `<Key>F4` specification at the bottom of Figure 51 to iconify the window when you press the **F4** function key:

```
<Key>F4        window          f.minimize
```

The mwm functions in the default key bindings section include:

- `f.post_wmenu`—Displays the window menu.
- `f.next_key`—Advances input focus to the next window. The `transient` option enables both primary client windows and their secondary windows to receive focus. This function will only work if `keyboardFocusPolicy` is set to `explicit`.
- `f.prev_key`—Advances input focus to the previous window. Refer to the previous bullet item for more information.
- `f.set_behavior`—Restarts mwm with the default behavior for your system.

Setting button bindings

In your .mwmrc file, you also set mouse button bindings to window manager functions. For example, the following line must be in your .mwmrc file to display the Root Menu when you press the left mouse button:

```
<Btn1Down> root f.menu RootMenu
```

The button bindings section of your .mwmrc file has the following syntax:

```
#comment
Buttons section_name
{
  [modifier_key] <button_action> context function
  [modifier_key] <button_action> context function
.
.
.
}
```

where

#	Indicates a comment line.
`Buttons` *section_name*	Specifies the name of the key bindings section.
{	Signals the beginning of the section's content.
modifier_key	Optional. Specifies that you must hold down either the **SHIFT** or **META** key while you perform *button_action*.
button_action	Specifies a mouse button and an action. For more information on button actions, refer to the mwm man page.
context	Specifies where the mouse pointer can be for *function* to occur. The possible values are `window`, `icon`, and `root`, `border`, `frame`, and `app`. To specify more than one context, separate them with a vertical bar (`|`).
function	Specifies one of the window manager functions. For more information, refer to the mwm man page.
}	Closes the button bindings section.

Figure 52 shows the default button bindings section in the system.mwmrc file.

Figure 52
Default button bindings section

```
# button binding descriptions
#
Buttons DefaultButtonBindings
{
     <Btn1Down>     icon|frame     f.raise
     <Btn3Down>     icon           f.post_wmenu
     <Btn1Down>     root           f.menu RootMenu
}
```

The default system.mwmrc file also has two alternate button bindings sections just below the default section. By default, mwm uses the key bindings section named `DefaultButtonBindings`. To use one of the alternate button binding sections, specify it by name in your .Xresource file. For example, to use `PointerButtonBindings` rather than the `DefaultButtonBindings`, you would enter:

```
Mwm*buttonBindings: PointerButtonBindings
```

in your .Xresource file. Again, to activate such a change, enter:

```
xrdb -load $HOME/.Xresources
```

and then choose the Restart... option from the Root Menu.

Specifying mwm resources

This section lists a few of the more useful mwm resources available with mwm. For a comprehensive list, refer to the mwm man page.

Resources that affect your .mwmrc file

As stated previously in this chapter, you must specify a few mwm resources in your .Xresources file in order for certain specifications in your .mwmrc file to work. These include:

- Any window menu except the `DefaultWindowMenu`. You can specify another window menu with:

  `Mwm*`*client_name*`*windowMenu:` *menu_name*

- Any key bindings except the `DefaultKeyBindings`. You can specify another key binding section in your .mwmrc with:

  `Mwm*keyBindings:` *key_binding_section*

- Any button bindings except the `DefaultButtonBindings`. You can specify another button binding section with:

  `Mwm*buttonBindings:` *button_binding_section*

To activate any of these changes, you must enter:

```
xrdb -load $HOME/.Xresources
```

and choose the Restart... option from the Root Menu.

Setting the focus policy

The mwm resource for setting the focus policy is `Mwm*keyboardFocusPolicy`. This resource has two possible values:

- `pointer`—Makes a window active when the mouse pointer is within its borders. This is the default for CXwindows, the CONVEX version of X Windows.
- `explicit`—Makes a window active when you click the left mouse button within the border of the window.

For example, to change the focus policy to explicit, enter the following resource specification in your .Xresources file:

```
Mwm*keyboardFocusPolicy:  explicit
```

Then, use the `xrdb` command to reload your .Xresources, and then restart mwm.

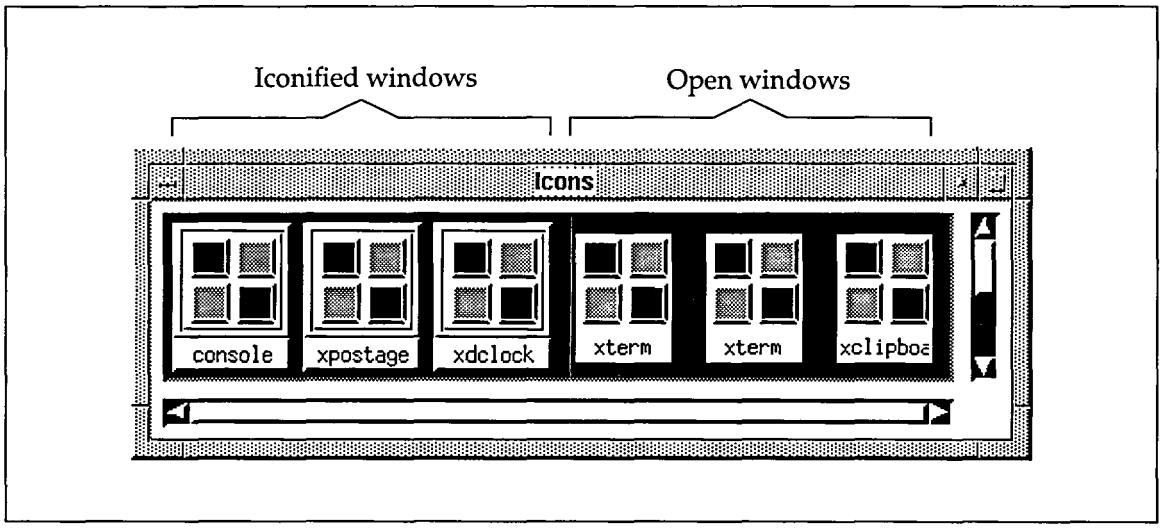
Using an icon box

If you want an icon box to help you manage your icons, place the following mwm resource in your .Xresources file:

```
Mwm*useIconBox: true
```

Figure 53 shows the mwm icon box. Notice that the three icons on the left represent iconified windows, while the three icons on the right represent open and active windows.

Figure 53
mwm icon box



Icons in an icon box work the same as they do without an icon box. To raise an iconified window, double click the icon. To display an icon's window menu, click the icon with the left mouse button.

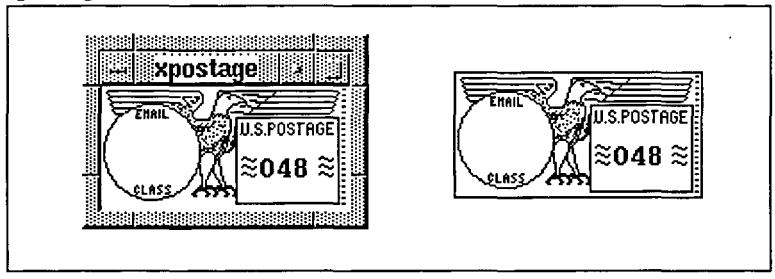
Removing client decoration

In some cases, you may prefer that a client not have a title bar or border. To remove parts of a client's border decoration, use the mwm `clientDecoration` resource. For example, to remove all border decoration from the xpostage client, you would enter the following resource specification in your .Xresources file:

```
Mwm*XPostage*clientDecoration: -all
```

Figure 54 shows the xpostage client with and without a border. Refer to the mwm man page for more information and other client decoration options.

Figure 54
xpostage client with and without a border



The `clientDecoration` resource also accepts the following values:

- `-border`—Removes the border but leaves the title bar.
- `-title`—Removes the title bar but leaves the border.

Removing icon decoration

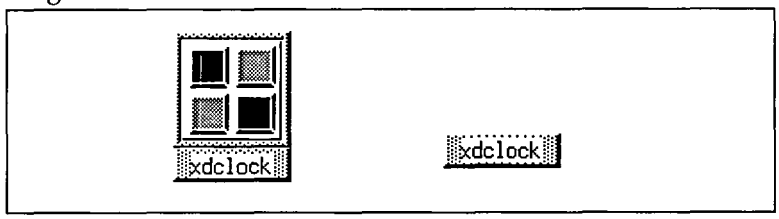
If you prefer to have smaller icons, you can also remove icon decorations by using the `iconDecoration` resource. For example, to make your icons the size of the label only, you would enter the following in your .Xresources file:

```
Mwm*iconDecoration: activelabel label
```

- `label`—Tells mwm to display the icon's label.
- `activelabel`—Tells mwm to enlarge the label when the mouse pointer is over the label.

Figure 55 shows the default icon on the left and the icon created by the previous resource specification on the right.

Figure 55
Large and small icons



Changing icon placement

With mwm, you can choose the location of your icons. By default, icons appear in the lower left corner. To change the location, use the `iconPlacement` resource. For example.

```
Mwm*iconPlacement: top right
```

places your icons in the upper-right corner in a column, and

```
Mwm*iconPlacement: left top
```

places your icons in the upper-left corner in a row.

Specifying fonts

5

A font is a style of typed characters. Most X clients allow you to choose the font that appears in menus, buttons, and text areas. This chapter teaches you how to:

- List available fonts and specify one
- Understand the parts of a font name
- Use font name aliases
- Specify fonts using wildcards
- View and select fonts with the xfontsel client
- Set your font search path
- Use scalable fonts
- Access the R5 font server

Listing available fonts with `xlsfonts`

To list all the font names that are currently available to you, enter `xlsfonts` at the command line. If you enter `xlsfonts | more` as shown in Figure 56, you can easily browse through the hundreds of fonts that `xlsfonts` lists.

Figure 56
Using `xlsfonts`

```
% xlsfonts | more
-adobe-courier-bold-o-normal--0-0-100-100-m-0-iso8859-1
-adobe-courier-bold-o-normal--0-0-75-75-m-0-iso8859-1
-adobe-courier-bold-o-normal--10-100-75-75-m-60-iso8859-1
-adobe-courier-bold-o-normal--11-80-100-100-m-60-iso8859-1
-adobe-courier-bold-o-normal--12-120-75-75-m-70-iso8859-1
-adobe-courier-bold-o-normal--14-100-100-100-m-90-iso8859-1
-adobe-courier-bold-o-normal--14-140-75-75-m-90-iso8859-1
-adobe-courier-bold-o-normal--17-120-100-100-m-100-iso8859-1
-adobe-courier-bold-o-normal--18-180-75-75-m-110-iso8859-1
-adobe-courier-bold-o-normal--20-140-100-100-m-110-iso8859-1
--More--
```

To use one of these fonts, you can use your mouse to copy the font name and paste it into a command at the command line or into your .Xresources file.

To specify a font for the xterm client from the command line, you would enter:

`xterm -fn` *fontname*

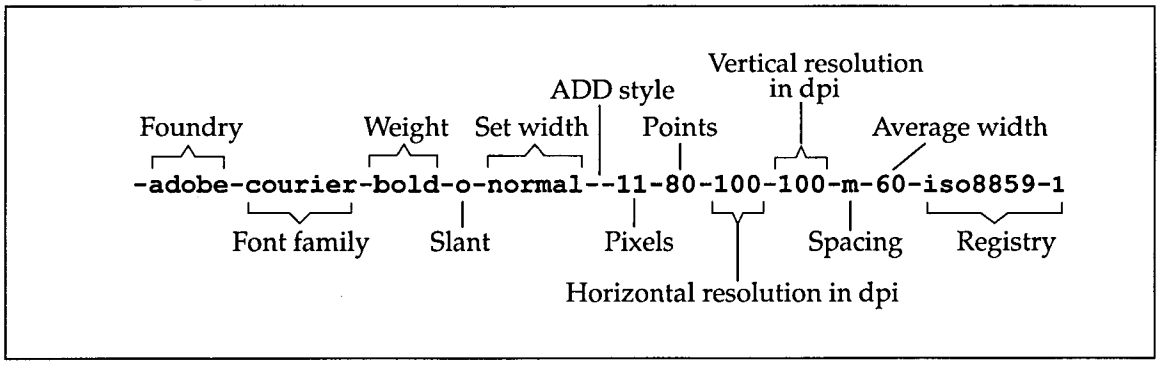
To specify a font for xterm in your .Xresources file, you would enter:

`xterm*font:` *fontname*

Deciphering font names

As you have seen, font names look fairly cryptic. Most font names are made up of the font's characteristics. Figure 57 shows the individual characteristics within a font name.

Figure 57
Font name desription



The font characteristics shown in Figure 57 include:

- **Foundry**—Lists the font manufacturer such as Adobe or Bitstream.
- **Font family**—Lists the style of the font such as Courier or Helvetica.
- **Weight**—Refers to how heavy the font is. Font weights include medium, bold, and demibold (medium bold).
- **Slant**—Indicates the type of slant that a font has. Slants include:
  - `r` for roman, which is upright
  - `i` for italic
  - `o` for oblique, which is similar to italic

- **Set width**—Describes the font's proportionate width. Currently, the only possible values are normal and semicondensed. You will rarely need to use set width. You should usually specify normal or an asterisk (*).
- **ADD style**—Indicates whether or not the font has serifs. A serif is the flourish on ascenders and descenders. The choices are `sans` for san serif (no serif) or nil. The nil value omits the characteristic. The -- in Figure 57 indicates the nil option.
- **Pixels**—Lists the size of the font in pixels. A pixel is a picture element: the smallest addressable unit of a bitmap screen.
- **Points**—Lists the point size of the font in tenths of a point.
- **Horizontal and vertical resolution in dpi**—Lists the resolution, usually 75 or 100. Use fonts with a dpi (dots per inch) of 100 for monitors that have a resolution close to 100 dpi. Use fonts with a dpi of 75 for monitors that have a resolution close to 75 dpi. To find out what your monitor's resolution is, use the `xdpyinfo` command and look at the resolution field.
- **Spacing**—Indicates that the font is either monospace (`m`), proportional (`p`), or character cell (`c`). The xterm client should use only use monospace or character cell fonts.
- **Average width**—Lists the mean width of all the characters in the font, measured in tenths of a pixel. You will rarely need to use average width. You can usually just specify *.
- **Registry**—Lists the registry and character set of the font.You can usually just specify *.

Using font name wildcards

Rather than looking through all the font names for a font that you want, you can use font name wildcards and specify only the font characteristics in a font name that are important to you.

There are two types of font name wildcards:

- * —You can replace any or all parts of a font name with an asterisk (*).
- ?—You can replace any single character with a question mark (?).

You can often get the font you want by specifying the font family, the weight, the slant, the point size, and wildcarding the rest. For example, suppose you want to use the xclipboard client with a Courier font that has no slant or bolding and has a point size of 100.

You could enter

```
xclipboard -fn '*courier-medium-r-*--*-100*'
```

The X server looks for this font in its font directories and uses the first font that has the specified characteristics.

There are a few things to keep in mind when using wildcards in font names:

- You must enclose the entire font name in quotes if you use a font name wildcard at the command line.
- If more than one font matches the name you have specified with wildcards, the server uses the first one it finds in alphabetical order. Because bold comes before medium and italic comes before roman, you will often get a bold, italic font if you just specify the font family.
- You can use wildcards in your .Xresources file; quotes are not necessary.
- If you specify characteristics that do not match any known fonts, the server uses a default font.

Using font name aliases

Some font names have aliases that are far shorter and easier to use than the actual name. These aliases are listed in a file called fonts.alias in each of the font directories in /usr/lib/X11/fonts. For example, /usr/lib/X11/fonts/misc/fonts.alias lists the aliases for the fonts in /usr/lib/X11/fonts/misc font directory.

The fonts.alias file

In the fonts.alias file, the alias is listed first, followed by the actual font name on the same line as shown in Figure 58.

Figure 58
Sample fonts.alias file

```
% more /usr/lib/X11/fonts/misc/fonts.alias
fixed -misc-fixed-medium-r-semicondensed--13-120-75-75-c-60-iso8859-1
variable -*-helvetica-bold-r-normal-*-*-120-*-*-*-*-iso8859-1
5x8 -misc-fixed-medium-r-normal--8-80-75-75-c-50-iso8859-1
6x9 -misc-fixed-medium-r-normal--9-90-75-75-c-60-iso8859-1
6x10 -misc-fixed-medium-r-normal--10-100-75-75-c-60-iso8859-1
6x12 -misc-fixed-medium-r-semicondensed--12-110-75-75-c-60-iso8859-1
6x13 -misc-fixed-medium-r-semicondensed--13-120-75-75-c-60-iso8859-1
6x13bold -misc-fixed-bold-r-semicondensed--13-120-75-75-c-60-iso8859-1
7x13 -misc-fixed-medium-r-normal--13-120-75-75-c-70-iso8859-1
7x13bold -misc-fixed-bold-r-normal--13-120-75-75-c-70-iso8859-1
--More--
```

You can use a font name alias the same way you use the regular font name. For example, if the following font is aliased to `5x7`:

```
misc-fixed-medium-r-normal--7-70-75-75-c-50-iso8859-1
```

then, rather than entering this large font name, you could use this same font by entering:

```
xterm -fn 5x7
```

Note

The fonts in the font directories come with your X server. Fonts and font name aliases may vary from server to server. If you do not have a fonts.alias file, font aliases are also listed by `xlsfonts`.

Creating your own font aliases

You can also create you own font name aliases by creating a file called fonts.alias in one of your own directories. This fonts.alias file uses the format shown in Figure 58. Place the alias first, followed by the font name on the same line:

```
myfont -*-*-medium-r-*-*-*-140-*-*-m-*-*-*
```

Then you must add the directory that this fonts.alias file is in to your font path with the `xset +fp` command. For example:

```
xset +fp /home/finkel/fontdir
```

In this case, fontdir is the directory that contains the fonts.alias file. For more information on changing your font path with the `xset fp` command, refer to "Setting the font search path" on page 79.

Now you can use your new alias as a regular font name:

```
xterm -fn myfont
```

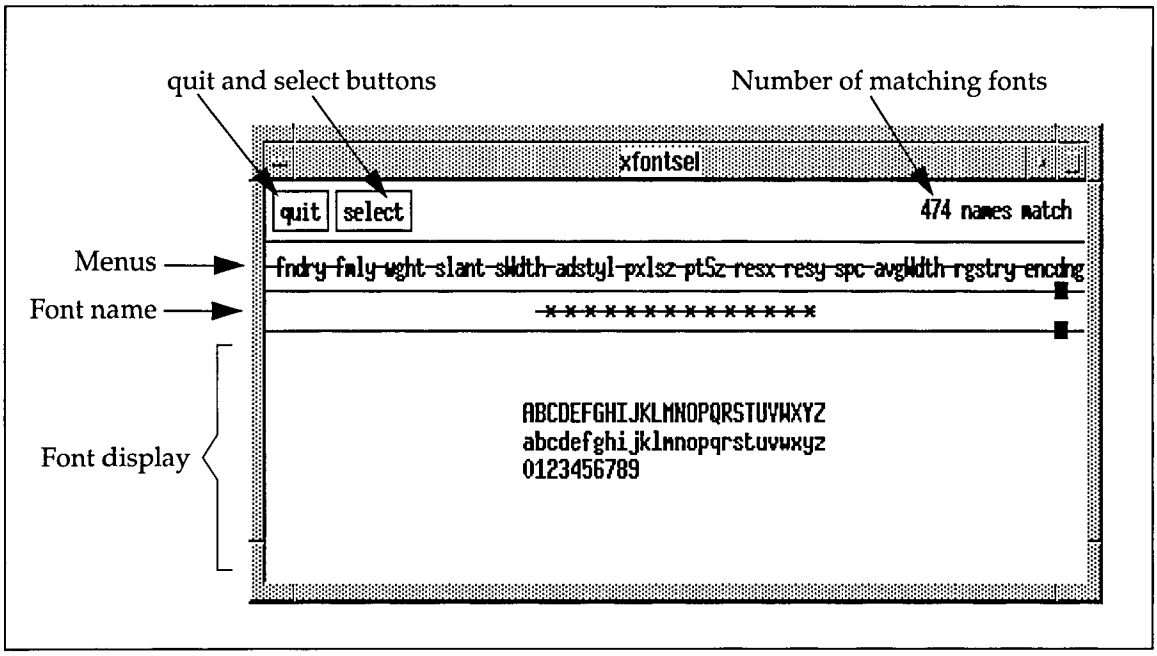
Choosing a font with xfontsel

The xfontsel client helps you choose a font by enabling you to preview fonts and then placing a font name in the cut buffer so that you can paste the font name with your mouse. To invoke the xfontsel client, enter

```
xfontsel &
```

from the command line. The window in Figure 59 appears.

Figure 59
The xfontsel client



The xfontsel window has the following features:

- **Menus**—Each name represents a menu that appears when pressed with the left mouse button.
- **Font name**—This area displays the font name (with wildcards) that matches the characteristics you have selected. The ? wildcard is not valid.
- **Font display**—This area displays the font that matches the characteristics you have selected.
- **quit button**—Quits xfontsel.
- **select button**—Places the displayed font name into the cut buffer, allowing you to paste the font name with your mouse.
- **Number of matching fonts**—Represents the number of fonts that have the characteristics you selected.

Using `xfontsel`

To use xfontsel, you select font characteristics from one of the menus. To use a menu:

1. Place the mouse pointer over a menu.
2. Press and hold down the left mouse button. The menu appears as shown in Figure 60.
3. While holding down the mouse button, move the mouse over the characteristic you want. The characteristic is highlighted.
4. Release the mouse button while your option is highlighted. Once you pick a characteristic, the font name changes, and the font display changes. This may take a while to complete.

Figure 60
xfontsel showing the family menu

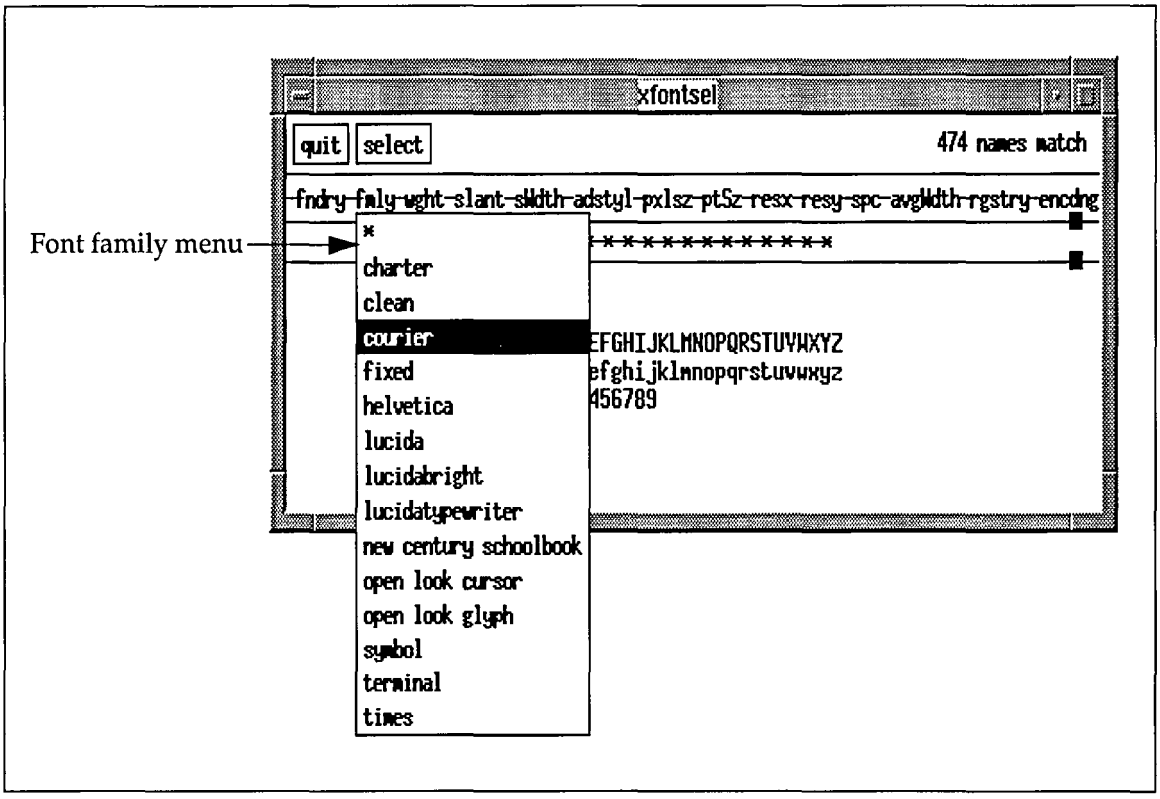
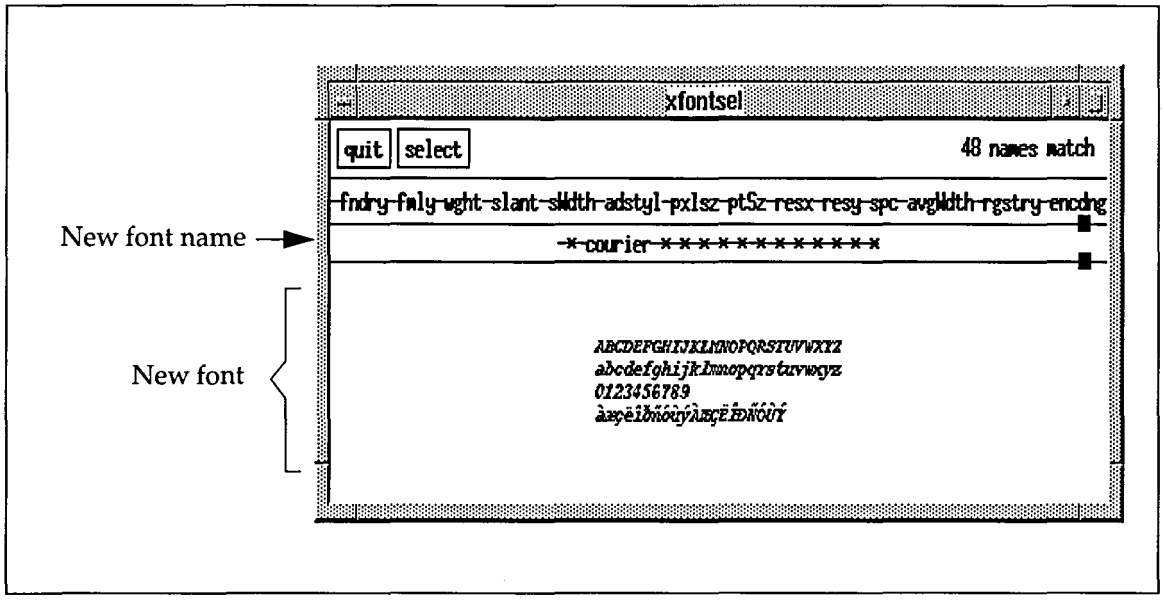


Figure 61 shows the results from selecting courier from the family menu. Notice that the font name and font display have changed.

Figure 61
Results of Figure 60



5. Continue to choose characteristics until a suitable font appears.
6. Click the select button with the left mouse button to place the displayed font name in the cut buffer. This enables you to paste the font name using the mouse.
7. Paste the font name into the command line or your .Xresources file by clicking the left mouse button.
8. Quit xfontsel by clicking the quit button with the left mouse button.

Setting the font search path

The font search path is a list of directories that your X server looks in to find fonts. By default, an R5 X server has the following font directories in its font search path:

- **/usr/lib/X11/fonts/misc**—Contains a variety of fixed width and symbol fonts.
- **/usr/lib/X11/fonts/Speedo**—Contains scalable fonts.
- **/usr/lib/X11/fonts/75dpi**—Contains fixed and variable width fonts that have a dpi of 75.
- **/usr/lib/X11/fonts/100dpi**—Contains fixed and variable width fonts that have a dpi of 100.

R4 X servers have all of these except the Speedo font directory. If you do not know what version of the X server you have, ask your system administrator.

You can add new directories to your font path with the `xset fp+` command. For example, if your system administrator tells you that there is new font directory in /usr/lib/X11/fonts/newfonts, you can add this directory to the end of your font path by entering the following command at the command line:

```
xset fp+ /usr/lib/X11/fonts/newfonts
```

You can place any of the `xset fp` commands in your X start-up script (either .xinitrc or .xsession). These commands include:

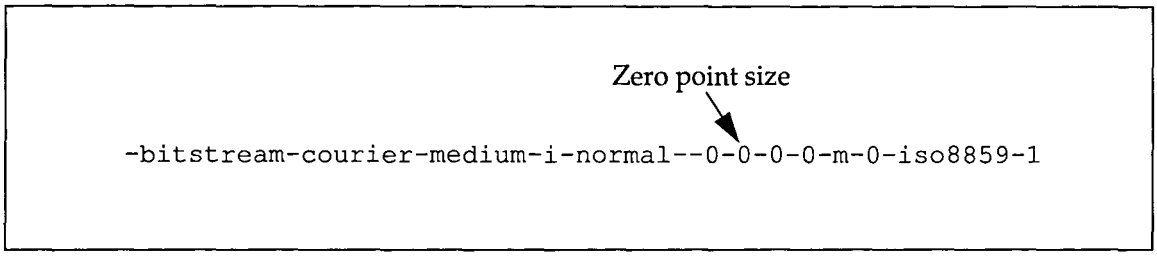
- `xset fp=` sets your font path.
- `xset +fp` adds a directory to the beginning of your font path.
- `xset fp+` adds a directory to the end of your font path.
- `xset -fp` deletes the first directory from your font path.
- `xset fp-` deletes the last directory from you font path.
- `xset fp default` resets your font path to the default.
- `xset q` lists the current `xset` setting, including the font path.

The changes to your font path happen immediately after you issue an `xset fp` command. However, the change is valid for the current session only. When you log back in, your server will use the default font path unless you place an `xset fp` command in your X start-up script.

Using scalable fonts

The R5 X server has many new fonts, including a directory of scalable fonts in /usr/lib/X11/fonts/Speedo. The font server (discussed in the next section) also has some scalable fonts. You can tell that a font is scalable if it has a point size of zero as shown in Figure 62.

Figure 62
Scalable font



A scalable font is a font that has a variable point size. This means that you can specify any point size when you use a scalable font. When you specify a scalable font, your X server scales the font to the specified point size rather than using a similar font with a different point size.

For example, suppose you like the following font:

```
-bitstream-courier-medium-r-normal--0-0-0-0-m-0-iso8859-1
```

You can make the point size much larger to allow a group of people to view text on your terminal by increasing its point size, in this case to 250:

```
-bitstream-courier-medium-r-normal--0-250-0-0-m-0-iso8859-1
```

Getting more fonts with the R5 font server

If you have an R5 X server and your host computer is running the font server program, then you have access to many more fonts via the font server. The font server is a central repository for fonts that is only available for R5 X servers.

To access fonts that reside in the font server, you must add the FONTSERVER environment variable to your shell configuration file and specify the name of your font server. For example, if you use the C shell, you would add the following line to your X start-up script:

`setenv FONTSERVER` *fontserververname*

If you use the Bourne shell, you would use the following:

`FONTSERVER=`*fontserververname*

```
export FONTSERVER
```

Ask your system administrator for the name of your font server.

To find out if you have access to the font server, use the `fsinfo` command as shown in Figure 63.

Figure 63
Using `fsinfo`

```
% fsinfo
name of server: earth:7000
version number: 1
vendor string: MIT X Consortium
vendor release number: 5000
maximum request size: 16384 longwords (65536
bytes)
number of catalogues: 1
 all
Number of alternate servers: 0
number of extensions: 0
```

If you do not have access to the font server, you will receive the following message:

```
fsinfo: unable to open server "(null)"
```

To list all the font names that are available through the font server, enter `fslsfonts` at the command line as shown in Figure 64. This command works like `xlsfonts` except that it lists fonts from the font server rather than fonts from your X server.

Figure 64
Using `fslsfonts`

```
% fslsfonts | more
-adobe-courier-bold-o-normal--0-0-100-100-m-0-iso8859-1
-adobe-courier-bold-o-normal--0-0-75-75-m-0-iso8859-1
-adobe-courier-bold-o-normal--10-100-75-75-m-60-iso8859-1
-adobe-courier-bold-o-normal--11-80-100-100-m-60-iso8859-1
-adobe-courier-bold-o-normal--12-120-75-75-m-70-iso8859-1
-adobe-courier-bold-o-normal--14-100-100-100-m-90-iso8859-1
-adobe-courier-bold-o-normal--14-140-75-75-m-90-iso8859-1
-adobe-courier-bold-o-normal--17-120-100-100-m-100-iso8859-1
-adobe-courier-bold-o-normal--18-180-75-75-m-110-iso8859-1
-adobe-courier-bold-o-normal--20-140-100-100-m-110-iso8859-1
--More--
```

Writing an X start-up script

6

It is tedious to start all of the clients you need every time you log in and start X. To save time and effort, you can write a start-up script that sets up your X environment automatically when you start X. This script can:

- Specify the name of your X server
- Start a window manager
- Load resources into your X server with `xrdb`
- Start X clients and position them on your screen
- Specify X preferences with `xset` and `xsetroot`

There are two types of X start-up scripts:

- **.xsession**—If you are using xdm, name your X start-up script .xsession in your home directory. xdm automatically executes the .xsession script in your home directory when you log in. If no .xsession file exists in you home directory, xdm uses a system default script.
- **.xinitrc**—If you are using a workstation without xdm, name your X start-up script .xinitrc[1] in your home directory. This script executes when you start X with the `xinit` command.

To customize your X start-up script, edit the script as described in the sections that follow.

Note

You must have execute permission for your X start-up script, or the script will not execute.

[1]Some workstations use a start-up script with a different name. Consult your workstation's documentation.

Setting the DISPLAY environment variable

The DISPLAY environment variable tells X the name of the X server to display clients on. This enables you to start a client from the command line without specifying the `-display` option. If you want the X client to appear on a different display, you can override the DISPLAY environment variable with the `-display` option from the command line.

If your system uses the X Display Manager (xdm), you do not need to set the DISPLAY environment variable because xdm sets it to the name of your X server when you log in.

If you are not running xdm, then the first line in your .xinitrc file should set your DISPLAY environment variable. If you are using the C shell, add:

```
setenv DISPLAY 'hostname':0
```

If you are using the Bourne or Korn shell, add:

```
DISPLAY='hostname':0
export DISPLAY
```

You must set your DISPLAY environment variable before you can start clients in your .xinitrc file.

Setting X preferences

Virtually every part of X can be customized, and your X start-up script is a good place to set many of your X preferences.

The most important preferences to set are the resource specifications in your .Xresources file. Place the following line in your X start-up script to load your resource specification:

```
xrdb -load $HOME/.Xresources
```

If you have used another name for your resource file, you should specify it here rather than .Xresources.

The `xrdb` command should appear before you start any clients in your script.

Using `xset` to set display preferences

This section lists many of the useful `xset` commands that enable you to set X display preferences. For more information, refer to the `xset` man page.

To view your current `xset` settings, enter `xset q` at the command line.

Setting the key click with `xset c`

You can turn the key click on, off, or specify a volume:

- `xset c off`—Turns key click off.
- `xset c on`—Turns key click on.
- `xset c` *n*—Sets key click volume. The maximum value is 100 and the minimum is 0.

Setting the bell volume with `xset b`

You can turn the bell on or off and change its volume, pitch, and duration:

- `xset b off`—Turns the bell off.
- `xset b on`—Turns the bell on.
- `xset b` *volume pitch duration*—Specifies integers to set these values.

Setting the font search path with `xset fp`

You can change your font search path with the following commands:

- `xset fp=` sets your font path.
- `xset +fp` adds a directory to the front of your font path.
- `xset fp+` adds a directory to the end of your font path.
- `xset -fp` deletes the first directory from your font path.
- `xset fp-` deletes the last directory from your font path.
- `xset fp default` resets your font path to the default.

For example, the following line adds a font directory to the front of your font path:

```
xset +fp /usr/lib/X11/newfonts
```

Setting mouse speed with `xset m`

You can change your mouse's acceleration and precision with the following command:

`xset m` *acceleration threshold*

where *acceleration* and *threshold* are integers.

The mouse will go *acceleration* times faster when it travels more than *threshold* pixels in a short time. This way you can use the mouse for precise alignment when you move it slowly. However, the mouse will move much faster when you move it quickly.

For example, `xset m 8 5` is a good setting that makes the pointer move very quickly when you move the mouse quickly and makes the pointer move more slowly when you move the mouse slowly.

With no parameters the `xset m` command resets the default. If you specify only one parameter, it is interpreted as acceleration.

Setting keyboard autorepeat with `xset r`

Autorepeat allows you to hold a key down and have the key's value repeat on the screen until you release the key. You can turn autorepeat on or off:

- `xset r off`—Turns autorepeat off.
- `xset r on`—Turns autorepeat on.

Setting the screen saver with `xset s`

You can set screen saver values with the following commands:

- `xset s` *timeout cycle*—The *timeout* value is the number of idle seconds before the screen saver is activated. The *cycle* value is the number of seconds before the screen saver pattern will change if you specify `noblank`.
- `xset s blank`—Tells the screen blank program to blank the screen rather than display a pattern.
- `xset s default`—Resets the default screen blank values.
- `xset s noblank`—Displays a pattern rather than blanking the screen.
- `xset s on`—Turns screen blanking on.
- `xset s off`—Turns screen blanking off.

Changing the root window with `xsetroot`

The `xsetroot` command enables you to change the appearance of the root (background) window. This section lists many of the more useful `xsetroot` options. For more information, refer to the `xsetroot` man page.

- `xsetroot -cursor` *cursorfile maskfile*—Enables you to specify a user-created cursor for the mouse pointer. The *cursorfile* is the bitmap file containing the cursor. The *maskfile* is the file containing the mask for the cursor. The mask is the outline of the cursor that enables it to show up when the cursor is over an area of the same color. Both files must be created with the bitmap X client.

- `xsetroot -cursor_name` *cursor_name*—Specifies one of the standard cursor names to use as a mouse pointer. For example:

  ```
  xsetroot -cursor_name spider
  xsetroot -cursor_name trek
  xsetroot -cursor_name gumby
  ```

  Standard cursor names are listed in the file /usr/include/X11/cursorfont.h.

- `xsetroot -bitmap` *filename*—Specifies the name of a file containing a bitmap to display on the root window. This bitmap can be made with the bitmap client, or you can specify one of the bitmaps in the directory /usr/include/X11/bitmaps.

- `xsetroot -mod` *x y*—Makes a grid pattern on the root window. The *x* and *y* parameters can be integers from 1 to 16.

- `xsetroot -grey`—Makes the root window grey.

- `xsetroot -solid` *color*—Makes the root window the color specified.

- `xsetroot`—Returns to the default values when you do not specify an option.

Starting X clients in your script

One of the most useful things to do in an X start-up script is to start the X clients that you always use and position them on your screen.

You should start all of the clients in your script in the background with the ampersand (&) except for the last one. The last client you start should remain in the foreground (without the ampersand) so that your start-up script will not end until you exit from the last client.

This last client should either be the window manager or an xterm window. If you start the window manager last, then you can end your X session by choosing the Exit option from the mwm Root Menu.

If you start an xterm window last, then you can end your X session by entering `exit` in the xterm window.

Specifying the size and screen position of a client

With the `-geometry` option, you can specify the size and location of a client. To specify the window size, enter the width and height as shown in the following example:

```
xterm -geometry 82x72 &
```

When the previous line is executed from the command line or from a script, it creates an xterm window that is 82 characters wide and 72 lines long. Most other X clients require that you specify the width and height in pixels. If you do not specify dimensions, the client uses a default value.

To specify a screen position with the `-geometry` option, you enter offsets from the edges of your screen:

- *+x+y*—Places the upper left corner of the client *x* pixels from the left side of the screen and *y* pixels from the top of the screen.
- *+x-y*—Places the lower left corner of the client *x* pixels from the left side of the screen and *y* pixels from the bottom of the screen.
- *-x+y*—Places the upper right corner of the client *x* pixels from the right side of the screen and *y* pixels from the top of the screen.
- *-x-y*—Places the lower right corner of the client *x* pixels from the right side of the screen and *y* pixels from the bottom of the screen.

Figure 65 graphically shows the offsets at each corner of the screen.

Figure 65
Corner offsets

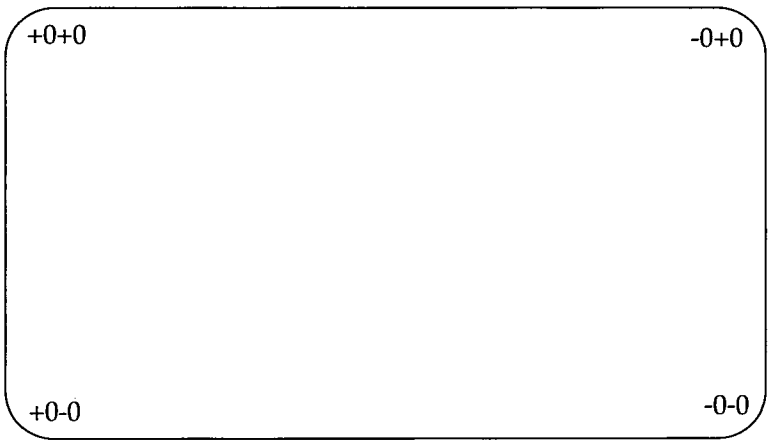


Figure 66 shows the location specified by the following command:

```
xterm -geometry 82x50+200+200 &
```

Figure 66
Location of +200+200

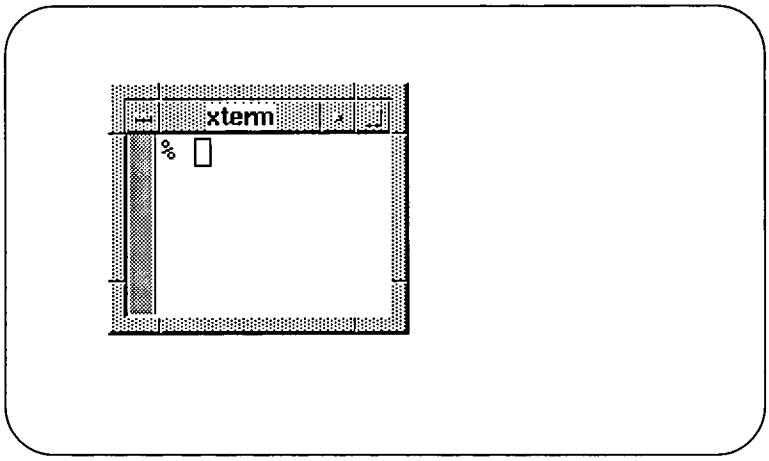


Figure 67 shows the location specified by the following command:

```
xterm -geometry 82x50+150-0 &
```

Figure 67
Location of +150-0

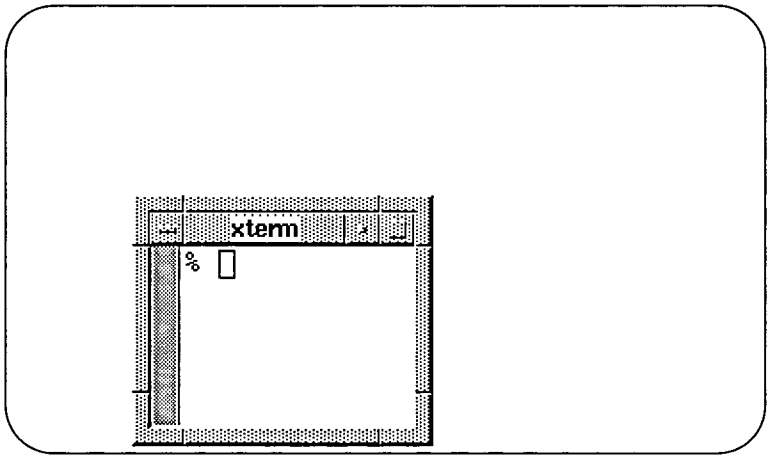
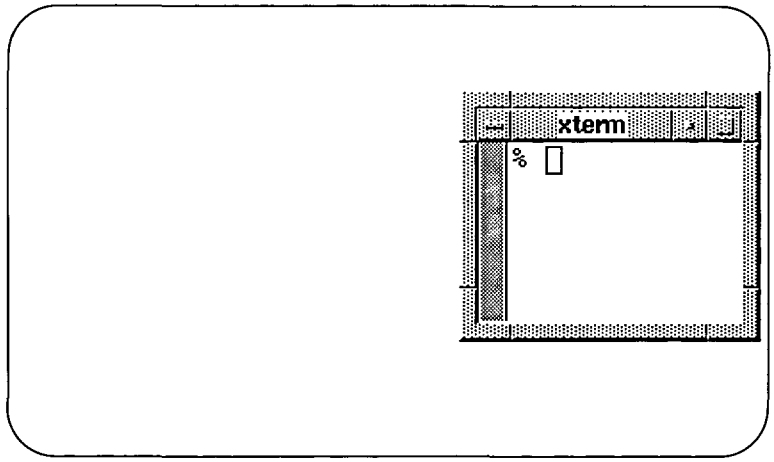


Figure 68 shows the location specified by the following command:

```
xterm -geometry 82x50-0+150 &
```

Figure 68
Location of -0+150



Finding a window's geometry with `xwininfo`

The `xwininfo` command lists the current geometry of a window on your screen. To use `xwininfo`:

1. Using the mouse, move a client to where you want it to appear when you start X.
2. Enter `xwininfo` at the command line. The client responds with:

```
xwininfo: Please select the window about which
you would like information by clicking the
mouse in that window.
```

3. Move the mouse pointer into the client you moved in step 1 and press one of the mouse buttons. `xwininfo` lists the information shown in Figure 69. The last two lines list the offsets from the corners and the window's geometry. You can paste the geometry line directly into a line in your X start-up script.

Figure 69
xwininfo output

```
% xwininfo

xwininfo: Please select the window about which you
 would like information by clicking the
 mouse in that window.

xwininfo: Window id: 0x140000d "xterm"

 Absolute upper-left X: 11
 Absolute upper-left Y: 28
 Relative upper-left X: 0
 Relative upper-left Y: 0
 Width: 757
 Height: 1084
 Depth: 1
 Visual Class: StaticGray
 Border width: 0
 Class: InputOutput
 Colormap: 0x21 (installed)
 Bit Gravity State: NorthWestGravity
 Window Gravity State: NorthWestGravity
 Backing Store State: NotUseful
 Save Under State: no
 Map State: IsViewable
 Override Redirect State: no
 Corners: +11+28 -832+28 -832-168 +11-168
 -geometry 82x72+0+0
```

Starting clients on remote computers

If you want to start clients in your script that run on a computer other than your host, you can do this with the `rsh` command. For example, the following line starts an xterm client on the computer named ralph:

```
rsh ralph -n exec xterm -display $DISPLAY -geometry 82x50+840+90 &
```

The `rsh` command connects to the computer named ralph and starts the `exec xterm` command.

When you run a client on a remote computer, you must specify the display. Rather then typing in your actual display name, you can enter `$DISPLAY` so that your script will work if you are using a different X display.

Starting the window manager

The most important X client to start in your script is the window manager. You may want to start mwm last in the foreground. This will allow you to end X with the Exit option from the mwm Root Menu. To do this, place the following line at the end of your X start-up script:

```
exec mwm
```

However, you may wish to start mwm in the background of the shell and start an xterm last in the foreground:

```
mwm &

(start clients)

exec xterm
```

This allows you to end X by entering `exit` in this xterm window.

Sample X start-up scripts

This section lists two sample X start-up scripts: one for use with xdm (.xsession) and one for use with a workstation (.xinitrc).

Sample .xsession script

The script in Figure 70 is a Bourne shell, X start-up script for an X display using xdm. The line numbers are used for reference purposes only and should not appear in your script. Each line should actually start in column one.

The following list describes each line in Figure 70.

1. Indicates that this is a Bourne shell script.
2. Redirects standard output and standard error to $HOME/.xsession.errors.
3. Adds the computers listed to the list of computers that are allowed to display clients on this X server. For more information on security, refer to Chapter 7, "Securing your X server."
4. Turns `xhost` security on.
5. Enables this X server to display clients running on the computer named venus. For more information on `xauth`, refer to Chapter 7, "Securing your X server."

6. Loads the resources in the .Xresources file.
7. Sets mouse speed and precision.
8. Starts a clock. The `-geom` option is an abbreviation for `-geometry`.
9. Starts the xbiff client on the host computer.
10. Starts an xterm on the host computer. The `-n` is an abbreviation for `-name`.
11. Starts another xterm on the host computer.
12. Starts the xload client on the computer named venus.
13. Starts mwm in the foreground. The window manager is started last so that X will end when you select Exit from the mwm Root Menu.

Figure 70
Sample .xsession script

```
1  #!/bin/sh

   # xdm is running, so do not set DISPLAY

   # Redirect standard out and standard error to a file.
2  exec > $HOME/.xsession.errors 2>&1

   # Set up security
3  xhost +astro mercury venus earth
4  xhost -
5  xauth extract $DISPLAY | rsh venus xauth merge

   # Load resource file
6  xrdb -load $HOME/.Xresources

   # Set mouse speed and precision
7  xset m 8 5

   # Start clients
8  xclock -geom 100x100-166-12 &

9  xbiff -geom -0-0 &
10 xterm -d $DISPLAY -geom 82x65+840+90 -ls -n earth -rv &
11 xterm -d $DISPLAY -geom 82x72+0+0 -ls -n earthxterm &
12 rsh venus exec xload -display $DISPLAY -geom -357-14 &

   # Start the window manager
13 exec mwm
```

Sample .xinitrc script

The script in Figure 71 is a C shell, X start-up script for a workstation that is not using xdm. The line numbers are used for reference purposes only and should not appear in your script. Each line should actually start in column one.

The following list describes each line in Figure 71.

1. Indicates that this is a C shell script.
2. Sets the DISPLAY environment variable to the workstation's name. By using `'hostname':0` rather than the actual display name, this script will work on any workstation.
3. Adds the computers listed to the list of computers that are allowed to display clients on this X server. The `>& $HOME/.xinit.errors` redirects xhost's response rather than letting it appear on the screen.
4. Turns `xhost` security on. For more information on security, refer to Chapter 7, "Securing your X server."
5. Loads the resources in the .Xresources file.
6. Sets mouse speed and precision.
7. Starts mwm in the background of the shell on the workstation. The `-display unix:0` reduces overhead when executing clients locally on a workstation.
8. Starts a clock that executes on the workstation. The `-geom` option is an abbreviation for `-geometry`.
9. Starts the xbiff client on the computer named earth.
10. Starts an xterm on the computer named earth. The `-n` is an abbreviation for `-name`.
11. Starts another xterm on the computer named earth.
12. Starts the xload client on the computer named venus.
13. Starts an xterm in the foreground on the workstation. This client is started last so that X will end when a user enters `exit` in this window. The `-C` option[1] indicates that this window should receive console output. The `-iconic` option starts this client as an icon.

[1]Some computers do not support the `-C` option.

Figure 71
Sample .xinitrc script for a workstation

```
1  #!/bin/csh -f

2  setenv DISPLAY `hostname`:0

   # Set up xhost security
3  xhost astro mercury venus earth >& $HOME/.xinit.errors
4  xhost - >>& $HOME/.xinit.errors

   # Load resource file
5  xrdb -load $HOME/.Xresources

   # Set mouse speed and precision
6  xset m 8 5

   # Start the window manager
7  mwm -display unix:0 &

   # Start clients
8  xclock -display unix:0 -geom 120x63+1268+1206 &

   # Start remote clients
9  rsh earth exec xbiff -display $DISPLAY -geom -0-0 &
10 rsh earth exec xterm -d $DISPLAY -geom 82x65+840+90 -ls -n earth -rv &
11 rsh earth exec xterm -d $DISPLAY -geom 82x72+0+0 -ls -n earthxterm &
12 rsh venus exec xload -display $DISPLAY -geom +1267+1101 &

   # Start local xterm in the foreground
13 exec xterm -C -n console -d $DISPLAY -geometry 80x18+840+0 -iconic
```

Securing your X server

7

Servers of any kind are usually available for anyone to use. For example, any user can print to a printer server. Your X server works the same way. This means that most anyone can access you X server to place windows on your screen or to watch your screen as you type text. This is certainly a security risk if the information on your screen is confidential.

This chapter explains how to:

- Use the `xhost` command to prevent other computers from accessing your X server.
- Use the `xauth` command to allow selected users to access your secured X server.

Using `xhost` to secure your X server

The `xhost` command enables you to prevent computers in your network from accessing your X server. With the `xhost` command, you can add and subtract computer names from a list of computers that are allowed to access your X server.

For example, the following command adds the computer named venus to the list of computers that can access your X server and removes the computer named mars:

```
xhost +venus -mars
```

When `xhost` access control is on, the previous command enables anyone on the computer named venus to access your X server and prevents anyone on the computer named mars from accessing your X server.

You turn `xhost` access control on by entering `xhost -`.

Enter `xhost` with no parameters as shown in Figure 72 to see the list of computers that are allowed to access your X server. The word "localhost" at the end of the `xhost` list in Figure 72 represents your workstation. X terminals do not have a local host.

Figure 72
Using `xhost`

```
% xhost
venus
earth
localhost
% xhost -
access control enabled, only authorized clients can
connect
```

You can use `xhost` in the following ways:

- `xhost` —Displays the names of the computers that are allowed to access your X server when `xhost` access control is on.
- `xhost +` —Turns access control off. This setting allows anyone on any computer to access your X server.
- `xhost -` —Turns access control on. This setting denies all computers access to your X server except those that you add to the list with the `xhost` *+hostname* command.
- `xhost` *+hostname*—Adds one or more computers to the list of computers that are allowed to access your X server. The plus sign is optional.
- `xhost` *-hostname*—Removes a previously added computer from the list of computers that are allowed to access your X server.

When you prevent a computer from accessing your X server, this means that X clients on that computer cannot access your X server. It also means that you cannot run X clients on that computer and display them on your screen.

Usually `xhost` is run from your X start-up script. You may also wish to use `xhost` from the command line to interactively allow access to other computers as needed.

The following example shows how `xhost` is used in an X start-up script:

```
xhost +earth venus mercury
xhost -
```

The first line adds the earth, venus, and mercury computers to the list of computers that can access your X server. The second line turns access control on, prohibiting all computers from accessing your X server except those in the list of allowed computers.

You can only issue `xhost` commands from your localhost if you are using a workstation without xdm. With xdm, you can issue `xhost` commands from any computer that has access to your X server.

Note

If you deny access to your local host or xdm host and you are not using the user authentication scheme described in the next section, then you will not be allowed to use the `xhost` command again until you log out and log back in.

The only way to block all users from accessing your X server using `xhost` is to deny access to all computers in your network, including your local host.

The `xhost` command does not terminate existing connections to your X server, so if you deny access to all computers, the X clients that are already on your screen will continue to have access to your X server.

For more information on security and `xhost`, refer to the `xhost` man page.

Using `xauth` with xdm

The X Display Manager (xdm) enables you to allow specific users access to your X server when you have denied access to all computers in your network with `xhost`.

To secure your X server with xdm, ask your system administrator to add the name of your display to the list of secured X servers.

How xdm security works

Once your system administrator makes your X server secure, xdm creates a .Xauthority file in your home directory when you log in. xdm puts an encrypted password (called a magic cookie) in this file. Each time you log in, xdm places a new encrypted password in your .Xauthority file.

If xdm security is enabled for your X server and you enter `xhost -` and remove all computers from the `xhost` access list, then no X client can access your display unless it knows the password in your .Xauthority file. This means that other users cannot place X clients on your screen or view the contents of your screen unless you explicitly allow them to.

However, if you have xdm security and enter `xhost +`, all users will be able to access your server whether they have the correct password or not.

Suppose you have a secured X server named cold:0. If user finkel starts an X client with the `-display` option set to cold:0, then this X client will not be allowed to appear on cold:0 unless a password in finkel's .Xauthority file matches the password in your .Xauthority file.

You should set the permissions on your .Xauthority file so that only you can read and write to it. This prevents other users from copying or deleting your password.

Your .Xauthority file can contain passwords for other servers besides your own. If other users give you their passwords, you can add them to your .Xauthority file with the `xauth add` command as described in the section that follows.

Allowing specific users to access your X server

You can send your .Xauthority password to selected users via email. To do this, enter the following command:

`xauth - list $DISPLAY | mail` *userid*

The user receiving this mail would then enter:

`xauth add` *info*

where *info* is the information sent via email. For example, if the name of your X server is cold and you want to allow user finkel to access your X server, you would do the following:

```
xauth - list $DISPLAY | mail finkel
```

When finkel looks at the mail message, he will see something similar to:

```
cold:0 MIT-MAGIC-COOKIE-1 04e05080812624dd64af1cfa2eca5059
```

Finkel would then use his mouse to copy this line and paste it into the following command line:

```
xauth add cold:0 MIT-MAGIC-COOKIE-1 04e05080812624dd64af1cfa2eca5059
```

Now finkel can access cold:0. However, the next time you log in, the password in your .Xauthority file will change. So, finkel will not be able to access your X server unless you send him the new password.

Allowing clients to run on other machines

The following information is relevant only if your home directory on your host computer is not mounted on the other computers in your network. This means that you have separate home directories on the other computers in your network.

While xhost and xdm security keep other users from accessing your X server, they also prevent you from displaying clients on your screen that are running on other computers in your network. You can allow your own clients on other computers to appear on your screen with the xauth command.

Normally, you would place the following line in your X start-up script for each computer that you want to run clients on and display them on your screen.

```
xauth extract - $DISPLAY | rsh computer xauth merge -
```

For example, if venus and mars are computers in your network, you can allow your clients running on venus and mars to appear on your screen by placing the following lines in your X start-up script:

```
xauth extract - $DISPLAY | rsh venus xauth merge -
xauth extract - $DISPLAY | rsh mars xauth merge -
```

The extract - command copies the password to standard output. The password is then piped to the other computer where the merge - command takes the password from standard input and places it in the .Xauthority file of your home directory on the computer specified.

In your X start-up script, you should place xauth commands in the foreground before you start any clients.

For more information on security and xauth, refer to the xauth man page.

Glossary

A

accelerator
A keyboard shortcut for issuing a command. Window accelerators are usually listed in the window menu of a window. Accelerators are defined in your .mwmrc file.

active window
The window that has input focus. The active window is the only window that accepts keyboard input.

ampersand (&)
Used after a UNIX command to place the command in the background.

Example: `xterm &`

B

bitmap
A grid of pixels, each of which is black or white. A color bitmap is called a pixmap. The bitmap client enables you to edit a bitmap to create mouse pointers, icons, or background pictures.

bitmap display
A monitor that displays information in bitmap format.

border
See *window border.*

border decorations
Window borders and title bars that have special functions. For example, the Iconify and Maximize buttons are part of the border decorations.

C

class
See *widget class.*

click
Press and release a mouse button quickly.

client
See *X client.*

copy and paste
The ability to use the mouse to copy text in one window and place it in another window.

.cshrc
The configuration file for C shell. This file is usually where C shell users set their path and environment variables.

cut buffer
The buffer used to store characters selected with the mouse. A buffer is memory used to store temporary information.

CXwindows
The CONVEX version of the X Window System.

D

decorations
See *border decorations.*

display
Your X server. The `-display` command line option is used to specify the name of an X server to display an X client on.

display device
See *X display device.*

double-click
Press and release a mouse button twice, very quickly.

drag
Move the mouse with a mouse button pressed down. For example, with mwm you can move a window by pressing the left mouse button while the mouse pointer is over the title bar and dragging the mouse to a new location.

E

explicit focus policy
Enables you to select a window for input by moving the mouse pointer into a window and clicking the left mouse button.

F

focusing
Choosing a new active window. This makes a window the input focus window. You cannot type in a window unless it has input focus.

G

geometry
A specification for the size and location of a window. You can specify a geometry in your .Xresources file or from the command line using the `-geometry` option.

graphic display device
A workstation or X terminal that has a keyboard, pointer device, and a bitmap display monitor.

I

icon
A small window that represents a larger window. With mwm, you can iconify a window by pressing the window's iconify button.

iconify button
A button on the window border that iconifies the window when you press it with the left mouse button.

implicit focus policy
Enables you to select a window for input by moving the mouse pointer into a window.

input focus window
The active window in which you can type.

instance
See *widget instance*.

L

Loose binding
A general case resource specification that skips widgets within a widget hierarchy. A loose binding uses asterisks to show that any number of widgets in the widget hierarchy have been skipped.

M

magic cookie
An encrypted password that is stored in your .Xauthority file. Each secured X server that is using xdm has an associated magic cookie that is needed to access the server.

Maximize button
A button on the window border that makes the window the size of your full screen when you press it with the left mouse button.

menu
A small window that contains choices that you can select with the mouse.

META key
A logical key that is usually mapped to one or more special purpose keys on your keyboard such as ALT.

minimize
Iconify a window.

mnemonic
A one-letter abbreviation for a command in a menu. The mnemonics for the window menu are defined in your .mwmrc file.

Motif
A user interface style designed by the Open Software Foundation and used with the Motif Window Manager (mwm).

mouse
A pointer device connected to your X display that moves a pointer cursor on your screen.

mwm
The Motif Window Manager.

.mwmrc
A file in your home directory that contains mwm menu specifications, mouse button bindings, and key bindings.

P

path
An environment variable that you set in you shell configuration file that allows you to access commands in various directories without having to specify a complete path name.

pointer device
A device, often called a mouse, connected to your display terminal that enables you to move a pointer cursor on your screen. A pointer device usually has two or three buttons that you can click to select choices on your screen.

protocol
See *X protocol*.

R

remote computer system
A computer in your network that is not your local host or your xdm host.

resize border handle
The parts of a window border that you can stretch with the mouse to resize the window.

resource manager
The part of your X server that maintains a database of resource specifications.

resource
See *X resource.*

root window
The background window on which all windows appear.

rsh
The remote shell command. This command enables you to execute a command on one computer from the command line of another computer.

S

scrollbar
The scrollbar enables you to review text that has scrolled off of the window.

server
See *X server.*

shell
An interactive command processor that is the interface between the user and the operating system.

slider
Also known as a thumb, this is the part of a scrollbar that you move with the mouse to position scrolled text in your window.

system.mwmrc
The default .mwmrc file. If you do not have a .mwmrc file when you start mwm, mwm uses the file in /usr/lib/X11/system.mwmrc.

T

tight binding
A resource specification that is fully qualified and uses dots. The dots separate widgets that are next to each other in an X client's widget hierarchy.

title bar
Contains the window's title and enables you to move the window with the mouse.

triple-click
Press and release a mouse button three times, very quickly.

W

widget
The small functional elements that make up a window such as text areas, menus, buttons, and scrollbars.

widget class
A group of widgets that have similar characteristics. A widget class name usually starts with a capital letter.

Example: Commands is the class name for all buttons.

widget hierarchy
The tree structure that describes the connections of widgets in a window. An X client's widget hierarchy is useful when you want to create a resource specification for the client.

widget instance
An actual occurrence of a widget.

window border
The small borders that surround the windows on your screen.

window decoration
See *border decoration*.

window manager
An X client that enables you to move, iconify, or resize windows on your screen.

window menu
The menu on each window that appears when you press the window menu button. This menu enables you to select window commands such as moving, resizing or iconifying a window.

window menu button
A button on the window border that displays the window menu.

windows
Areas on your screen that represent application programs.

workstation
A stand-alone computer that has its own processor, memory, and possibly a disk drive.

X

X client
An X application program.

X defaults
Refer to *X resource*.

X display device
A workstation or an X terminal that can run the X server program and has a mouse, a keyboard, and a bitmap display.

X protocol
The mechanism that X clients use to communicate with X servers.

X resource
A variable that you set in your .Xresources file that changes the appearance or functionality of an X client. For example, the following resource specification makes the xterm client have a scrollbar:

```
xterm*scrollbar: true
```

X server
A program that runs on your display device and provides display and input capabilities for X clients.

X terminal
A graphic input and output device that must be connected to a host computer. X terminals have a built-in X server.

X Window System
A graphic windowing system designed for networked computer systems with bitmap displays.

`xauth`
A command used to manipulate the password in your .Xauthority file.

.Xauthority
A file that contains a password needed to access a particular X server.

xdm
The X display manager. This program manages a number of X servers in a network, provides a login prompt, and provides an authorization scheme for secured X servers.

`xdpyinfo`
A command that displays information about your X server such at its name and screen resolution.

`xhost`
A command that prevents other computers from accessing your X server.

`xinit`
The command used to start the X server on many workstations. This command also executes the .xinitrc X start-up script.

.xinitrc
The name of the X start-up file used by many workstations.

`xrdb`
The command used to load X resources in to the resource manager.

Example: `xrdb -load .Xresources`

.xsession
The name of the X start-up file that you use if your system is running xdm.

xterm
A terminal emulation program. Each xterm window contains a separate shell session.

Index

Symbols

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

X

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 ADDRESSEE

KEITH KNOX
CONVEX COMPUTER CORPORATION
PO BOX 833851
RICHARDSON TX 75083

Free books for your feedback!

Please help us provide better manuals by answering the questions on this form. No envelope or postage is necessary. Simply cut out this card, mail it, and we'll send you one to five free copies of this book!

Send me 1, 2, 3, 4, or 5 free copies of this book. (Circle one.)

Circle the answer that best expresses your opinion:

	Agree			Disagree		No Opinion
This manual has the information that I need.	5	4	3	2	1	0
This manual is easy to understand.	5	4	3	2	1	0
Information is easy to find.	5	4	3	2	1	0
There are enough examples.	5	4	3	2	1	0
The examples are easy to follow.	5	4	3	2	1	0
The illustrations are easy to understand.	5	4	3	2	1	0

What sections of this manual do you use most often?

_____

_____

What information should be added to this manual?

_____

_____

_____

Additional comments_____

_____

_____

_____

_____

Name _____ Date _____

Position _____

Department _____

Company _____

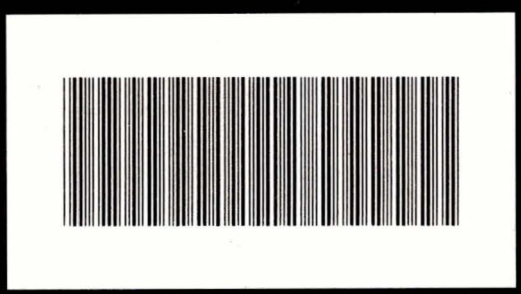
Address _____

City _____ State _____ Zip _____

May we contact you?_____ Phone_____

cut here

cut here



Order Number
DSW-218

Document Number
710-022630-000