HEWLETT PACKARD

# Using the X Window System

# Using the X Window System

HEWLETT
PACKARD

HP Part No. B1171-90064
Printed in U.S.A. August 1992

Edition 6

## Printing History

The manual printing date and part number indicate its current edition. The printing date will change when a new edition is printed. Minor changes may be made at reprint without changing the printing date. The manual part number will change when extensive changes are made.

Manual updates may be issued between editions to correct errors or document product changes. To ensure that you receive these updates or new editions, you should subscribe to the appropriate product support service. See your HP sales representative for details.

| | |
|---|---|
| December 1988 | Edition 2 |
| September 1989 | Edition 3 |
| February 1991 | Edition 4 |
| November 1991 | Edition 5 |
| August 1992 | Edition 6 |

Hewlett-Packard Company
User Interface Technology Division
1000 N.E. Circle Blvd.
Corvallis, OR 97330

# Contents

# Figures

# Tables

# 1

# Introduction

Welcome to the X Window System version 11 (X11 or X). The X Window System is a network transparent window system.

The HP Visual User Environment (HP VUE) is a graphical user interface that is based on the X Window System. The X Window System can be run alone, or as part of HP VUE. This manual covers what is needed to run the X Window System by itself, although there is a lot of valuable information for the HP VUE user as well.

In this chapter you'll find out how this manual is organized and some of the conventions it uses.

## Who Should Read this Manual

The primary audience for this manual is system administrators for systems running the X Window System *but not* HP VUE. However, HP VUE users who want information on the font server should read chapter 6, "Using Fonts."

Since HP VUE provides other mechanisms for performing some of the actions covered in this manual, HP VUE users should first look in the *HP Visual User Environment User's Manual.*

## How This Manual Is Organized

| | |
|---|---|
| Chapter 1 | Introduction. Gives some tips, and describes other documentation available to you. |
| Chapter 2 | Hardware and software that are part of a typical X11 system and explains general concepts. |
| Chapter 3 | Configuration information for default file, multiple screens, remote operation, special input devices, Starbase applications, and Native Language support. |
| Chapter 4 | Starting, using, and stopping X. |
| Chapter 5 | How applications obtain resources. |
| Chapter 6 | How and where to use different fonts. |
| Chapter 7 | Motif Window Manager. |
| Chapter 8 | X11 clients. |
| Chapter 9 | Special mouse and keyboard configurations. |
| Chapter 10 | Printing and screen dumps. |
| Chapter 11 | Starbase graphics. |
| Reference | "man pages"—reference for X clients. |
| Appendix B | Using the Keyboards. |
| Glossary | Special terms. |

## Conventions

As you read this manual, notice the following typographical conventions:

**Table 1-1. Typographical Conventions**

| If you see ... | It means ... |
|---|---|
| `computer text` | This text is displayed by the computer or text that you type exactly as shown. For example,<br>`login:`<br>is a login prompt displayed by the computer. |
| *italic text* | A book title, emphasized text, or text that you supply. For example,<br>`hpterm -fg` *color*<br>means you type "hpterm -fg" followed by a color you choose. |
| ▢ | You press the corresponding key on the keyboard. For example,<br>CTRL Left Shift Reset<br>means you hold down the CTRL key, the Left Shift key, and the Reset all at the same time. |
| `[ ]` | An optional parameter that can be left off if you don't need that functionality. For example,<br>`xload [-rv] &`<br>means that you must type "xload" but don't have to type "-rv". |
| `{ }` | A list containing *mutually exclusive* optional parameters. For example,<br>`xset r` $\left\{\begin{array}{l}\texttt{on}\\ \texttt{off}\end{array}\right\}$<br>means that option `r` can be set to either on or `off`, but not both. |
| **bold text** | The definition of this term follows. Often the term is also defined in the glossary. |

Also, you can use the X Window System with either a two- or a three-button mouse by observing the following conventions. These are the default mouse button settings and can be changed as described in chapter 9.

**Table 1-2. Mouse Buttons and Their Locations**

| **If you see ...** | **On a 2-button mouse press ...** | **On a 3-button mouse press ...** |
|---|---|---|
| Button 1 | The left button. | The left button. |
| Button 2 | Both buttons simultaneously | The middle button. |
| Button 3 | The right button. | The right button. |

Be careful of your spelling:

- Watch uppercase and lowercase letters. A file named `.xdefaults` is *not* the same file as `.Xdefaults`. Use uppercase letters where indicated and *only* where indicated.
- Don't confuse the number 1 (one) with the letter "l" (el).
- Don't confuse the "0" (zero) with the upper case "O" (oh).
- White space (extra spaces or tabs) at the end of a command line in a text file sometimes alters the meaning of the command. Files such as `.rhosts` are especially vulnerable. After modifying a file, check for unwanted white space.

## For More Information

Read these books to find out more about HP-UX, HP VUE, text editing, OSF/Motif, widgets, and widget programming:

| **Title** | **HP Part Number** |
|---|---|
| *Using HP-UX With HP VUE* | B2910-90001 |
| *Shells User's Guide* | B2355-90046 |
| *The Ultimate Guide to vi and ex Text Editors* | 97005-90015 |
| *HP Visual User Environment User's Guide* | B1171-90061 |
| *OSF/Motif Programmer's Guide*, Prentice Hall, Englewood Cliffs, NJ | B1171-90060 |
| *Mastering OSF/Motif Widgets* by Donald L. McMinds. Addison-Wesley Publishing Company, Reading MA:1992 | B1171-90057 |

Finally, depending on your needs, the following books about the X Window System might prove useful:

- *Introduction to the X Window System* by Oliver Jones. Prentice Hall, Englewood Cliffs, NJ:1989.
- *The Definitive Guides to the X Window System Volume Three: X Window System User's Guide for Version 11 Release 5* by Tim O'Reilly and Valerie Quercia. O'Reilly and Associates, Petaluma, CA:1992.

2

# What is the X Window System?

This chapter describes:

- Basic concepts.
- The hardware and software of a typical system.
- Distributed computing.

## Basic Concepts

This section introduces several fundamental concepts:

- The role of the X server.
- Multi-tasking environments.
- Distributed computing.

### The Server-Client Interaction Model

The X **server** usually starts during system boot before the login screen is displayed. The display server controls all access to input devices (typically the mouse and keyboard) and all access to output devices (typically the display). You can think of it as standing between the programs running on your system and your system's input and display devices.

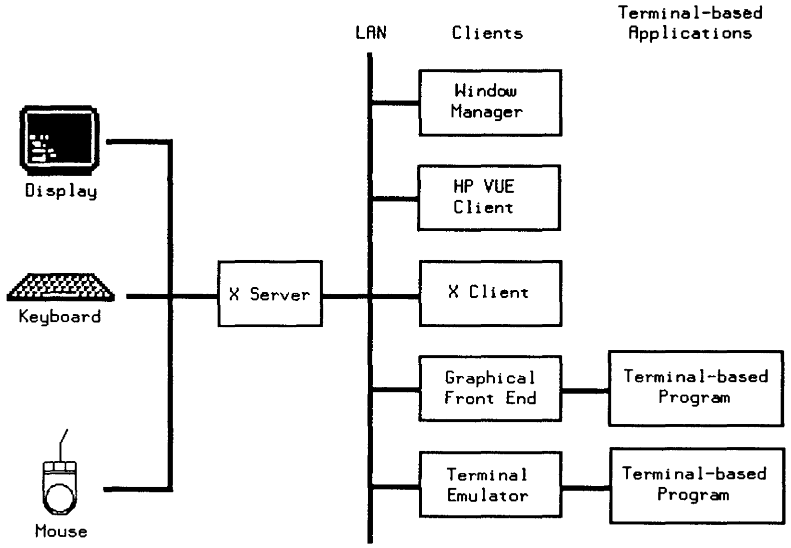


**Figure 2-1. The Server Controls Display Access.**

A **client** is any program written to run with the server. Clients know about windows and workspaces and how to make use of them. **Non-clients** are programs that don't know how to make use of windows.

## Multi-Tasking

Multi-tasking is the computer's ability to execute several programs simultaneously. Each program is a separate task (process). Each process usually runs in a separate window, and processes running in separate windows do not interfere with one another. For example, you can have the system recalculate a large spreadsheet in one window while you shift your attention between editing a monthly report in a second window and answering your electronic mail in a third. Each program normally has a main window for visual interaction, and each window has its own input and output.

Only one window at a time receives user input. That window is called the **active window**. While you focus on one window, other windows continue running unattended or wait for your input.

## Distributed Computing—Local and Remote Access

### Remote Access

Networked computing environments provide the ability to run programs on computers other than the one you are sitting in front of. For example, you can run a program locally and display the output on the screen of a remote system. Conversely, you can run a program remotely and display the output in a window on your screen. You can also run a program remotely and have it display on yet another remote screen.

### The Distributed Computing Environment

A Distributed Computing Environment (DCE) is a group of computer systems joined together into a network. Resources resident on one system are available to all systems. A LAN (Local Area Network) provides the link.

X11 doesn't care where a program is—it simply communicates to the program via the LAN connection. This structure permits you to operate workstations at a strictly local level with all client programs residing locally, or at a networked level with some programs running locally and others running remotely.

In addition, other systems on the LAN can run programs that reside on your workstation and direct the visual output to *any* screen on the network.

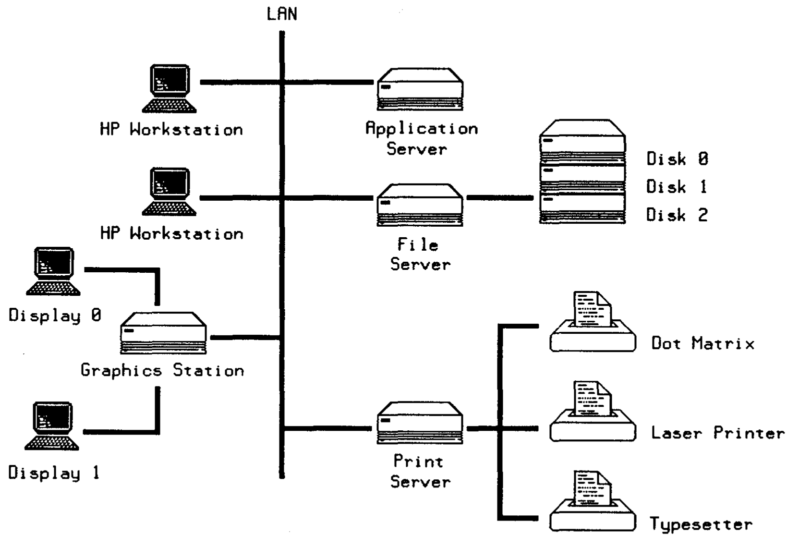


**Figure 2-2. A Typical Distributed Computing Environment.**

## Application Servers

One of the workstations in figure 2-2 is an **application server**. An application server provides the processing power and memory necessary to run large processor-intensive applications. A user at one of the workstations can log into the application server to enter necessary data and run complex programs. The output can be directed to a window on the user's workstation. More than one user can run programs at the same time.

## File Servers

Another workstation in figure 2-2 is a **file server**. A file server controls the storage and retrieval of data from hard disks. The file server provides a central location for files, which means that less storage space is required on an individual's local computer. It also provides a relatively inexpensive and quick backup facility.

A file server can also serve as a hub for diskless workstations. You can have a **cluster** of several diskless workstations connected to a single hub with a large disk. Each workstation, or **node**, needs a certain amount of individual space on the disk, but all nodes can share the system and application software, eliminating the need for redundant local system storage.

## Print Servers

One of the workstations in figure 2-2 has several printers attached to it and acts as a **print server**—a computer that controls spooling and other printing operations. Page formatters and page composition programs reside on the print server and are invoked with the proper commands.

## Graphics Stations

Certain applications use graphics accelerators to speed up the presentation of graphics on the screen. Generally, engineers working with CAD (Computer-Aided Design) applications are the major users of graphics accelerators. Hewlett-Packard supports graphics acceleration with a graphics library called Starbase. The graphics station in figure 2-2 has two high-performance graphics subsystems attached to it. Each subsystem is powerful enough to run the X Window System while running a Starbase application.

The graphics station permits a larger number of people to share the expensive hardware and software resources required by a CAD/CAM station. Tasks that engineers may have that do not require graphics acceleration can be accomplished at their desks on a more typical workstation.

## Multi-Vendor Communications

DCE allows computers manufactured by different vendors, running different operating systems, to communicate with each other over the LAN. If you are using a computer made by Hewlett-Packard, you can communicate over the LAN directly with computers made by a variety of other manufacturers supporting X, as long as each is running the X Window System and connected to a LAN using the Ethernet protocol standard.

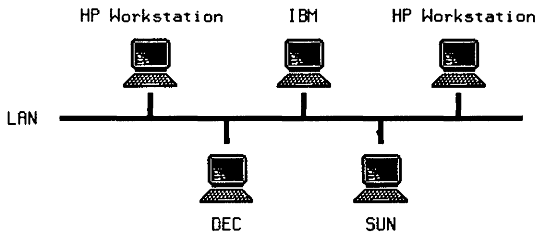


**Figure 2-3. Multi-Vendor Communication.**

## The Parts of a Typical X11 System

All X11 systems have the following features in common:

- Computer hardware.
- The operating system.
- An X server program to control communication between the display and client programs.
- Client programs, including:
  - A window manager to control the display's window environment.
  - Application programs to provide useful services.

### Hardware

The hardware system consists of several components:

#### System Processing Unit (SPU)

The **SPU** contains the logic circuitry that performs all the processing that takes place. The SPU runs the server, takes care of foreground and background processing, and controls local and remote accessing of your system's resources.

#### The Hard Disk

The hard disk stores programs and data files. Some configurations are called **diskless clusters** because groups of users share the same hard disk.

#### Keyboard

The keyboard is an **input device** used to type information into the computer. Although the keyboard is frequently used in conjunction with a mouse, it does not need to be. You can configure X11 so that you can use the keyboard for both text entry (its usual purpose) and for pointing and selecting (the mouse's usual purpose). **Mouseless operation** may be beneficial in situations where desk space is at a premium.

**Note** There are now two keyboards available for Hewlett-Packard workstations, the 46021 keyboard, and the C1429 keyboard. See appendix B, Using the Keyboards, for more information on using these keyboards and the differences between them.

### Mouse and Other Pointing Devices

A pointing device lets you point to a specific area on the screen and select it. A mouse is the most common pointing device. Mouse movements and button presses can be associated with keyboard key presses for mouseless operation.

The server also supports other pointing devices—for example a digitizer tablet or track ball. References to mouse actions apply also to corresponding actions with other devices.

### Display

The display is the principal output device. A typical display consists of one physical screen per mouse and keyboard. However, a display can include as many as four physical screens, all using the same mouse and keyboard.

The screen becomes the **root window** when you boot X11. The root window contains all the windows, menus, and icons that comprise the visual elements of your X11 environment.

Technically, the screen is known as a **bitmapped device** because the graphical elements (windows and icons) that it displays are stored by the computer as a **bitmap**, a pattern of bits (dots) that can be readily displayed as graphical images.

### Local Area Network (LAN)

The LAN is composed of hardware and software. The hardware connects the computer system physically to a network that includes other computer systems at your site and could connect to other networks at different locations. The LAN enables you to take advantage of remote processing capabilities of X11.

## Software

There are several types of software that comprise the X Window System.

To an end-user, the layers blend together into a single working environment. However, from a system administration point of view, it is important to know how the layers work together.

### The Operating System

The operating system is the software that controls the operation of the computer system. The X Window System runs on the Hewlett-Packard HP-UX operating system. This is a multi-user, multi-tasking environment. A multi-user environment means more than one user can be on the system at the same time. A multi-tasking environment means that each of those users can run more than one program at a time.

### The X Server

The central part of the X Window System is the **server**, also called the **X server** or **display server**. The server is the program that controls the screen, keyboard, and mouse, and processes communication requests. The server updates the windows on the screen as a client generates new information or as you enter information through an input device. All client programs communicate through the server.

### The Font Server

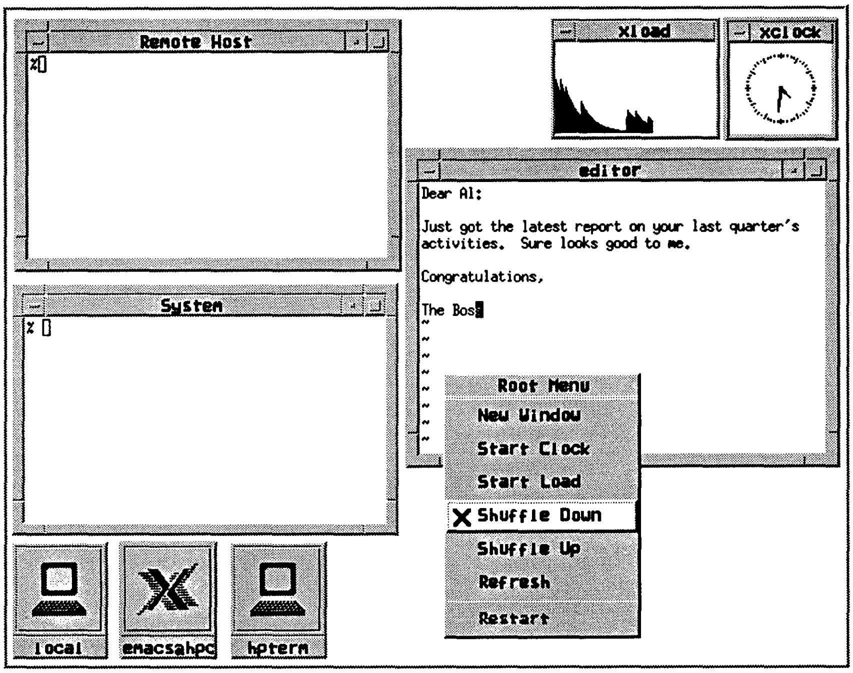
The font server allows a font administrator to distribute fonts to all X servers in a system from a central point. The font server is covered in chapter 6.

### The Window Manager

The window manager is your main means of dynamically controlling the size, shape, state (icon or normal), and location of the windows on your screen. It also supplies the frames and menus for the windows.

The window manager is the first client started during a session after the X server has started. All other clients with their own windows must be able to interact with the window manager.

This manual covers the OSF/Motif Window Manager (`mwm`). Using the window manager is covered in chapter 4. Configuring the window manager is covered in chapter 7.



```
Dear Al:

Just got the latest report on your last quarter's
activities.  Sure looks good to me.

Congratulations,

The Boss
```

**Figure 2-4. Windows, Clients, Menus, and Icons**

## X Clients

Clients are programs designed to run under the X Window System.

There are a number of clients that are included with the X Window System. For example, the `xrdb` client provides the ability to view and modify current resources.

Some clients (for example, `xwininfo` and `xmodmap`) do not create windows. They use an existing terminal emulation window to display their output.

Clients are discussed in chapter 8.

## Non-Client Programs

Non-client programs are designed to run alone on display screens or "terminals" and are therefore referred to as **terminal-based** programs. Terminal-based programs must have terminal emulator windows created for them so that they can run in a window environment.

# 3 Preliminary Configuration

This chapter covers some of the preliminary configuration you may need to do before starting the X server. It includes:

- Setting the `DISPLAY` environment variable.
- Using hardware and software configuration files.
- Using custom screen configurations.
- Configuring the system for special input devices.
- Distributed processing.
- Using Native Language Support.

There are other chapters that deal with initial configuration for special situations:

- Chapter 4 covers starting and running X.
- Chapter 6 covers configuring and running the font server.
- Chapter 7 covers configuring the window manager.
- Chapter 11 covers configuration for running the X Window System with Starbase.

## Do You Need to Read This Chapter?

All users should check:

- the `DISPLAY` variable.
- the `X0.hosts` file.
- the `/etc/hosts` file if your system is not configured to query a nameserver.

The rest of this chapter covers optional configuration. The following table shows the assumed configuration, and what you should read if you want to change it.

**Table 3-1. Default X Configuration**

| Expected configuration | If you want to change it, read ... |
|---|---|
| 1 display | "Using Custom Screen Configurations" |
| 1 mouse | "Using Special Input Devices" |
| 1 keyboard | "Using Special Input Devices" |
| American English language | "Customizing for Native Language Support" |
| X starts the `hpterm` and `mwm` clients as part of its own start-up procedures. | "Software Configuration Files" chapter 4 |
| Default `mwm` colors, window decorations, and menus. | "Software Configuration Files" chapter 5 chapter 7 |
| Font server not started | "Using the X11R5 Font Server" chapter 6 |

## Finding Your System Directory

The directory containing most of the X Window System configuration files is called the *system directory*. It is `/usr/lib/X11`

## Setting the DISPLAY Variable

The `DISPLAY` environment variable establishes the host, display number, and screen number to which a system sends bitmapped output.

You can check the current setting of your system's `DISPLAY` variable by typing this command:

```
env
```

A list similar to the following is displayed:

```
DISPLAY=hpaaaaa:0.0
HOME=/users/ellen
TZ=PST8PDT
:
```

The `DISPLAY` variable has the syntax:

$$\left[\left\{\begin{array}{l} \textit{hostname} \\ \texttt{local} \\ \texttt{unix} \\ \texttt{shmlink} \end{array}\right\}\right] : \textit{display[.screen]}$$

The default is *hostname*`:0.0`, which is display 0, screen 0 of the display running the X server.

To reset the `DISPLAY` environment variable type the appropriate command shown below, or put it into the configuration file used by your system if you want it to be in effect every time you log in.

**Table 3-2. Setting Environment Variables**

| Shell | Command | Configuration File |
|---|---|---|
| sh | `DISPLAY=`*host:display.screen*<br>`export DISPLAY` | `~/.profile` |
| csh | `setenv DISPLAY`<br>*host:display.screen* | `~/.login` |
| Aegis | `DISPLAY :=` *host:display.screen*<br>`export DISPLAY` | `~/user_data/startup_dm.`*xxx* or<br>`/sys/dm/startup_login.`*xxx* |
| ksh | `DISPLAY=`*host:display.screen*<br>`export DISPLAY` | `~/.profile` |

## Making an X0.hosts File

The `/etc/X0.hosts` file is an ASCII text file containing the hostnames of each remote host permitted to access your local server.

- If you are running as a stand-alone system, you must have your system's name in this file.
- If you are part of a network, the other system names must be included.

The syntax is as follows:

*host*
*host*
*host*

For example, if you are `hpaaaaa`, and regularly ran clients on `hpccccc`, and `hpddddd`, you would want the following lines.

```
hpaaaaa
hpccccc
hpddddd
```

Note that aliases work as well as hostnames, provided they are valid, that is, commonly known across the network.

The default screen configuration file `X0screens` uses the default X11 remote host file `X0.hosts`.

Each custom `X*screens` file is associated with a special `X*.hosts` file. The number represented by the `*` causes the correct screen and host files to be used together. For example, `X3screens` takes an `X3.hosts` file. Both are referenced by the server when it is started with a `/usr/bin/X11/X :3` command.

If you use a special `X*screens` file, you need to set your `DISPLAY` variable appropriately. For the previous example, it would be set to `hostname:3.0`.

## Using an /etc/hosts File

This file need not be present if your system is configured to query a nameserver.

The `/etc/hosts` file is an ASCII text file containing a list of all the host names and internet addresses known to your system, including your own system.

If your system is not connected to a network, use the loopback address (`127.0.0.1`) and the host name `unknown`:

```
127.0.0.1  unknown
```

For a local system to access a remote host:

- The address and hostname of the romote host must be listed in the local system's `/etc/hosts` file.
- The user must have a valid login (username and password) and home directory on the remote host.

## Software Configuration Files

The X Window System uses four configuration files:

| | |
|---|---|
| `.Xdefaults` | Specified default appearance and behavior characteristics for clients. The contents of this file are covered in more detail in chapter 5. |
| `.x11start` | Specifies the clients that start when the X Window System starts. This file is covered in more detail in chapter 4. |
| `.mwmrc` | Specifies the menus, menu selections, button bindings, and keyboard bindings that control the OSF/Motif Window Manager (`mwm`). The contents of this file are discussed in chapters 5 and 7 |
| `app-defaults/*` | Optional configuration for specific clients. The contents of this file are discussed in chapter 5. |

If your home directory does not contain these files, the X Window System uses the system-wide versions of these files in your system directory:

```
sys.x11start

system.mwmrc
```

If you want to customize your X environment, copy these files from the system directory to your home directory (noting the name change), and make your modifications. For example:

```
cp /usr/lib/X11/sys.mwmrc $HOME/.mwmrc
cp /usr/lib/X11/sys.Xdefaults $HOME/.Xdefaults
```

The X server looks first in your home directory for these files. If they are not there, it uses the system-wide files.

## Using Custom Screen Configurations

The default screen configuration is specified in the `X0screens` file in your system directory. It assumes:

- There is one display—display 0.
- There is one screen—screen 0.
- The screen uses only one set of pixel planes (the image planes).
- The screen is at the address node specified by `/dev/crt`.

If you use some configuration other than the default, you must edit the default screen file or add additional screen configuration files.

There should be a separate `X*screens` file for each display, where * is a number that matches the display number used when starting the X server.

### Creating a Custom 'X*screens' File

There are two ways to create a custom screen configuration for a display:

- You can modify `X0screens` so that it contains device information for all the screen configurations you may want to use. This is generally the preferred way. Only one configuration is used at a time; the others are commented out. To switch from one screen configuration to another, you uncomment some lines and comment others. For multiple displays, you would have a separate file for each display—for example, `X1screens` for display 1.
- You can have a separate `X*screens` file for each screen configuration on a particular display. Switching between them involves modifying the command that starts the X server.

## Screen Modes

The syntax of the lines in `X*screens` depends on the screen mode you choose. Depending on your system's hardware, you may choose from four screen modes:

3

| | |
|---|---|
| image mode | The default screen mode using multiple image planes for a single screen. The number of planes determines the variety of colors available to the screen. |
| overlay mode | An alternate screen mode using overlay planes for a single screen. Overlay planes provide an alternate (auxiliary) set of planes to the standard image planes. You can see what is in the image planes only if you open a "transparent" window in the overlay planes and move the window over what you want to see. Typically, overlay planes are used in conjunction with image planes in either stacked mode or combined mode. |
| stacked mode | A combination of image and overlay planes in which a single display has two "logical" screens: image planes and overlay planes. You move between the image and overlay planes by moving the pointer to the edge of the display. |
| combined mode | A combination of image and overlay planes in which a single display has a single screen that is a combination of the image and overlay planes. Typically, Starbase and other display-intensive applications run in image-plane windows that can be moved and resized like any other window, but the window frames are maintained in the overlay planes. |

Table 3-3. Display Hardware and Screen Modes

| With this display hardware ... | You can use these modes ... | | | |
|---|---|---|---|---|
| | Image | Overlay | Stacked | Combined |
| HP A1096 | √ | | | |
| HP A1416 | √ | √ | √ | |
| HP A1439 | √ | | | |
| HP A1659 | √ | | | |
| HP A1924 | √ | | | |
| HP A1991 | √ | | | |
| HP 9000S300 model 425e | √ | | | |
| HP 9000S700 model 710 | √ | | | |
| HP 98542A | √ | | | |
| HP 98543A | √ | | | |
| HP 98544B | √ | | | |
| HP 98545A | √ | | | |
| HP 98547A | √ | | | |
| HP 98548A | √ | | | |
| HP 98549A | √ | | | |
| HP 98550A | √ | √ | √ | |
| HP 98704A | √ | √ | | √ |
| HP 98720A | √ | √ | √ | |
| HP 98730A | √ | √ | √ | √ |
| HP 98735A | √ | √ | | √ |
| HP 98765 | √ | √ | | √ |

## Syntax of 'X*screens'

Each line in `X*screens` lists a separate screen device (except in combined mode). A screen device can be a physical device, the CRT screen, or the image planes or overlay planes of a physical device.

The syntax for each line of an `X*screens` file is:

$$\texttt{/dev/}\mathit{device}\left[\begin{array}{l}\left\{\begin{array}{l}\texttt{depth}\left\{\begin{array}{l}8\\24\end{array}\right\}[\texttt{doublebuffer}]\\ \texttt{depth 16 doublebuffer}\end{array}\right\}\\ \texttt{monitorsize }\mathit{number}\end{array}\right][\texttt{\# }\mathit{comment}]$$

| | |
|---|---|
| /dev/*device* | Specifies the name of the device file that the X server should read for this display. |
| depth | Specifies the number of image and overlay planes available to the server (one pixel per plane). |
| doublebuffer | Specifies **double buffer**. Double buffering divides video memory into halves and displays one half while drawing the other. Double buffering is used with graphics programs that double buffer their screen output. This avoids "flashing" during screen redraw. |
| depth 16 doublebuffer | Specifies the division of the image planes into two 8-bit, double buffered halves. |
| monitorsize | Specifies the size of the monitor in inches (when *number*$\leq$100) or millimeters (when *number*>100). |

### Example

This example shows how to specify a particular screen configuration consisting of a high-resolution screen on which you want to run X11 and Starbase applications. The image plane of this screen is accessed by the device file `/dev/`*image_device*. The overlay plane is accessed by the device file `/dev/`*overlay_device*. You want to switch between four different screen configurations:

- One screen with X11 running in the image planes (image mode).
- Two screens with Starbase running in the image planes and X11 running in the overlay planes (overlay mode). You may have only one Starbase application running in this mode, and you can see it only if you open a "transparent" window to look through the overlay planes.
- Two screens running on the same display (stacked mode). One screen runs in the image planes, and the other runs in the overlay planes. You move between the two screens by moving the pointer to the edge of the display. The order in which the screens appear is specified by the order in which their designations appear in the `X*screens` file. Starbase is not normally run in this mode.
- One screen with two visuals, one with depth 24, and the other with depth 8 (combined mode). Starbase applications are run in windows that can be moved or resized like any other window. You can have several Starbase applications running at once, each in its own window. The order in which the screens are described is unimportant.

Here is the `X0screens` file that creates these four configurations. To use a particular configuration, uncomment the line (two lines for Stacked Screens Mode) that create it, and make sure all other lines are commented.

```
### Default Configuration ###
/dev/image_device

### Overlay Screens Mode ###
# /dev/overlay_device

### Stacked Screens Mode ###
# /dev/image_device
# /dev/overlay_device

### Combined Screens Mode ###
# /dev/image_device /dev/overlay_device depth 24 depth 8
```

More examples can be found in the system `X0screens` file. Note that the first configuration is the default screen configuration provided with X11.

## Multiple Screen Configurations

On systems that support multiple display devices, it is possible to configure X to use these multiple devices in two ways:

- A single X server can treat multiple devices as individual screens within a single display.

  Configure the X*screens file so that each device is listed on a separate line. The order of the lines in the file determines the number given to each screen, beginning with zero.

  ```
     /dev/crt0     #display 0, screen 0
     /dev/crt1     #display 0, screen 1
  ```

  To address a specific device, use the screen parameter of the DISPLAY variable. For instance, DISPLAY=local:0.1 addresses the second device.

- Multiple X servers can treat each device as a separate display.

  Create an X*screens file for each device. For example:
  X0screens:

  ```
     /dev/crt0    #display 0, screen 0
  ```

  X1screens:

  ```
     /dev/crt1    #display 1, screen 0
  ```

  To address a specific device, use the display parameter of the DISPLAY variable. In this case, DISPLAY=local:1.0 addresses the second device.

## Mouse Tracking with Multiple Screen Devices

If you use a multi-screen configuration, the mouse pointer can move from one screen to another. You can arrange the screens in a vertical, horizontal, or matrix orientation by adding the appropriate lines to the X*pointerkeys configuration file described in chapter 9. The sample X*pointerkeys file in your system directory contains examples that show how to specify the orientation of multiple screens.

**Note** The sample X*pointerkeys file is placed in /usr/lib/X11 at install time. If you subsequently update your system, the X*pointerkeys file in /usr/lib/X11 is *not* overwritten, and the sample file is placed in /etc/newconfig.

Moving the mouse pointer off one edge of a screen causes the pointer to move to another screen, depending on the screen orientation you have specified. configuration files, the order of entry determines the tracking order of the mouse pointer. The first line in the file is the device on which the pointer appears when you start X11.

Other lines correspond to the screens that appear when the mouse is moved to the right or left side of the current screen. Moving off the right side goes to the next higher display, the left side the to next lower. If you are on the highest display and move right, you move to the lowest display. If you are on the lowest display and move left, you move to the highest display.

## Making a Device Driver File

Devices specified in screen configuration files must correspond to device files. If you don't have the appropriate device file, you must create it using the `mknod` command. For information on `mknod` see the system administration manual for your operating system.

## Using Special Input Devices

The X server reads an input device file, `X0devices` in your system directory, to find out what input devices it should open and attach to the display.

### The Default 'X0devices' File

The default `X0devices` file contains lines of text, but does not specify any input configuration. Rather, it assumes the default input configuration of one keyboard and one pointer.

If this is your configuration, you may not want to change the contents of the file for three reasons:

- Clients can request and receive the services of an input device regardless of whether the device is specified in a device configuration file. Thus, you need not change the `X0devices` file, or create a custom file, even though you have a custom input configuration.
- Non-clients (terminal-based programs) such as Starbase cannot receive the services of an input device if the device is specified in the device configuration file. Any device in the device configuration file is opened for use by the X server. Thus, changing the `X0devices` file or creating a custom file to inform the server about a certain input device may interfere with a non-client's ability to access the device.
- Even if you have other screen configurations, you can rely on the default input device configuration without having to create an `X*devices` file to match every `X*screens` file. For example, if you had a custom `X*screens` file, you would not necessarily need an `X*devices` file.

A custom `X*devices` file is required only when you want to tell the X server about a custom input device configuration.

## How the Server Chooses the Default Keyboard and Pointer

Input devices attach to HP-UX computers through an interface known as the Hewlett-Packard Human Interface Link (HP-HIL). Up to seven input devices can be attached to each HP-HIL. However, if the `X*devices` file does not exist, or does not specify otherwise, the X server recognizes only two devices: a pointer and a keyboard (clients, however, may still recognize other input devices).

The X server uses the following order when choosing a pointer or keyboard:

1. If the `X*devices` file specifies an input device as the pointer or keyboard, the X server uses that device as specified.
2. If `X*devices` makes no specification, or there is no `X*devices` file, the X server takes the last mouse on the HP-HIL (the mouse farthest on the link from the computer) as the pointer and the last keyboard (the keyboard farthest on the link from the computer) as the keyboard.
3. If the X server can open no mouse, it takes the last pointer device (knob box, graphics tablet, trackball, or touchscreen) on the HP-HIL as the pointer.
4. If the X server can open no pointer device, it takes the last keyboard on the HP-HIL as the pointer as well as the keyboard.
5. If the X server can open no keyboard, it takes the last key device (buttonbox, barcode reader) on the HP-HIL as the keyboard.
6. If no pointer and keyboard can be opened, the server will not run, unless it is explicitly instructed to run without a keyboard or pointer by the `X*devices` file.

## Using Serial (RS-232) Input Devices

You can use some serial input devices with the X Server. A *device driver* must exist for the desired serial input device, and it must reside in the directory `/usr/lib/X11/extensions`. You must also modify the `X*devices` file (found in the directory `/usr/lib/X11`) to inform the X Server which serial input device is to be used, the serial port to which it is connected, and how it is to be used (as the X pointer, X keyboard, or accessed through the input device extension).

You can determine which serial input devices are configured by examining the `/usr/lib/X11/X0devices`. For example, assume a Lighted Programmable Function Keyboard is connected to the computer via the first serial port. This keyboard is an extra device that client programs access through protocol requests defined by the X input extensions. It can be identified to the X Server by adding the following lines to the X*devices file:

```
Begin_Device_Description
Name  lpfk.sl
Path  /dev/tty00
Use   extension
End_Device_Description
```

## Creating a Custom 'X*devices' File

At some point, you may want to instruct the server to open a particular device as the keyboard or pointer or have the server open another input device and merge its input keyboard or pointer. Additional devices with keys are merged with the keyboard; additional devices that point merged with the pointer.

To tell the server about a non-default input device configuration, you must add a device specification line to the appropriate `X*devices`. For example, you would use `X0devices` if you used `X0screens` and `X2devices` if you used `X2screens`.

### Syntax for Device Type and Relative Position

The HP-UX operating system can refer to a device by its location on the HP-HIL. This syntax uses device type and relative position on the HP-HIL to specify input devices. Separate the part of your entry with tabs or spaces.

*relativeposition devicetype use* [# *comments*]

| | |
|---|---|
| *relativeposition* | Specifies the position of a device on the HP-HIL relative to other input devices of the same kind. |
| *devicetype* | Specifies the type of input device. |
| *use* | Specifies whether the device is the keyboard, the pointer, or has some other use. |

Valid positions, types, uses, and examples are in the section "Selecting Values for 'X*devices' Files" in this chapter.

Separate the parts of your entry with tabs or spaces.

The position of an input device on the HP-HIL is relative to other devices of that type. Thus, `first` means the device connected closest to the computer on the HP-HIL of any device of that type.

This syntax is most useful for systems running a single X server with no other programs directly opening input devices. Here, if you add a new input device to the HP-HIL, you don't have to edit the `X*devices` file unless the device is of the same type as one already named in the file and you add the new device ahead of that existing device.

This syntax may become ambiguous when more than one X server is running on the same system or when non-client programs directly access input devices. This is because `first` actually means first device of that type available to the server. Thus, a device may be physically first on the HP-HIL, but not first for the server if the device is unavailable because it is currently being used by some other program.

### Syntax for Device File Name

This syntax uses the device file name to specify input devices:

*/path/devicefile use* [# *comments*]

| | |
|---|---|
| */path/devicefile* | Specifies the path and device file to use as the input device. |
| *use* | Specifies whether the device is the keyboard, the pointer, or has some other use. |

This syntax is unambiguous when several X servers are running on the same computer or when non-client programs directly access the input device.

### The Syntax for Reconfiguring the Path to Device Files

The default path to the device files is `/dev/hil`, but you can specify another path if you choose. Also, if you have more than one HP-HIL, you can specify which HP-HIL the server should use.

The syntax is:

*path* `hil_path` [# *comments*]

| | |
|---|---|
| *path* | Specifies the path to the device files. |

The X server appends numbers to the path to create the names that it attempts to open for use as input devices. For example, specifying

```
/tmp/foo hil_path
```

results in the device names `/tmp/foo1`, `/tmp/foo2`, and so on.

3

## Selecting Values for 'X*devices' Files

`X*devices` files use the following special names for positions, devices, and uses:

**Table 3-4. Values for 'X*devices' Files.**

| Positions | Device Type (Device Class) | Uses |
|---|---|---|
| first | keyboard (keyboard) | keyboard |
| second | mouse (pointer) | pointer |
| third | tablet (pointer) | other |
| fourth | buttonbox (keyboard) | |
| fifth | barcode (keyboard) | |
| sixth | one_knob (pointer) | |
| seventh | nine_knob (pointer) | |
| | quadrature (pointer) | |
| | touchscreen (pointer) | |
| | trackball (pointer) | |
| | null | |

The nine-knob box appears to the X server as three separate input devices. Each row of knobs is a separate device with the first device being the bottom row.

Note also that the HP barcode reader has two modes: keyboard and ASCII. The modes are set via switches on the reader. If you set the barcode reader to ASCII transmission mode, it appears to the server as a barcode reader and the device name is therefore `barcode`. However, if you set the barcode reader to emulate a keyboard, the barcode reader appears as a keyboard and the device name should therefore be `keyboard`. What distinguishes a barcode reader set to keyboard mode from a real keyboard is the relative position or the device file name, depending on which syntax you use.

Similar to the barcode reader, the trackball appears to the server, not as a trackball, but as a mouse. Therefore, to specify a trackball, use the `mouse` device name. Again, what specifies the trackball instead of the real mouse is the relative position or the device filename, depending on which syntax you use.

## Examples

You can create a system on which the X server runs, but which does not have any input devices. In this case, clients could be run from a remote terminal, or from a remote host, and their output directed to the X server. To create a system with no input, include the following lines in the `X0devices` file:

```
first null   keyboard
first null   pointer
```

If you had a more complicated configuration, such as two graphics tablets, two keyboards, and a barcode reader, your `X*devices` file could look like this:

```
first     tablet     pointer     The pointer.
second    tablet     other       Merged with the pointer.
first     keyboard   other       Merged with the keyboard.
second    keyboard   keyboard    The keyboard.
first     barcode    other       Merged with the keyboard.
```

In this example, the first tablet acts as the pointer, the second keyboard acts as the keyboard, input from the second tablet is treated as if it came from the X pointer, and input from the first keyboard and the barcode reader is treated as if it came from the X keyboard.

Note that the barcode reader is in ASCII mode in this example. If the barcode reader were in keyboard mode, the last line of the example should read as follows:

```
third keyboard    other
```

More examples can be found in the `X0devices` file in `etc/newconfig`.

## Customizing for Native Language Support (NLS)

This section covers:

- How X uses the LANG environment variable and other environment variables.
- Accessing language-dependent message catalogs and resource files.
- Remote execution in NLS systems.

### Setting the LANG Environment Variable

The LANG environment variable must be set in order to use native language support. Setting LANG causes X to use the language-sensitive routines for character handling.

You can set LANG to any value in the /usr/lib/nls/config file. The X Window System does not currently support the arabic or hebrew languages. The message catalogs are located in subdirectories of /usr/lib/nls.

The LANG environment variable is set in the same way as the DISPLAY environment variable discussed earlier in this chapter.

**Table 3-5. Supported Languages**

| | | | |
|---|---|---|---|
| american | american.iso88591 | bulgarian | C |
| c-french | c-french.iso88591 | chinese-s | chinese-t |
| czech | danish | danish.iso88591 | dutch |
| dutch.iso88591 | english | english.iso88591 | finnish |
| finnish.iso88591 | french | french.iso88591 | german |
| german.iso88591 | hungarian | icelandic | icelandic.iso88591 |
| italian | italian.iso88591 | japanese | japanese.ujis |
| katakana | korean | norwegian | norwegian.iso88591 |
| polish | portuguese | portuguese.iso88591 | romanian |
| russian | serbocroatian | spanish | spanish.iso88591 |
| swedish | swedish.iso88591 | | |

No suffix on the language name implies HP roman8 coding.

## Other NLS Environment Variables

This section covers other NLS environment variables. It provides an overview only. For detailed information, refer to *X Toolkit Intrinsics Programming Manual.*

### Message Catalogs—The NLSPATH Environment Variable

The `NLSPATH` environment variable determines the paths applications search for NLS message catalogs. X clients place NLS message catalogs in the `/usr/lib/nls/$LANG` directories. Both `LANG` and `NLSPATH` must be set in order to use those message catalogs.

It shouldn't be necessary to set this variable unless the message catalogs are installed in non-standard locations.

The proper value of `NLSPATH` depends on whether message catalogs exist for the current value of `LANG`.

To use the message catalogs for the language to which `LANG` is set, set `NLSPATH` to:

```
/usr/lib/nls/msg/%L/%N.cat:/usr/lib/nls/C/%N.cat:$NLSPATH
```

### Setting the XUSERFILESEARCHPATH Environment Variable

The `XUSERFILESEARCHPATH` environment variable controls where X applications look for their `app-defaults` resource files. The default location is in directory `/usr/lib/X11/app-defaults`. You need to set `XUSERFILESEARCHPATH` if your resource files are not in this location.

For example, to use Japanese `app-defaults` you would set `XUSERFILESEARCHPATH` to `/usr/lib/X11/japanese/app-defaults`.
Or, you could set `XAPPLRESDIR` to `/usr/lib/X11/%L/app-defaults` and `LANG` to "japanese". If `LANG` is not set, `%L` defaults to null.

If you set `XUSERFILESEARCHPATH` in `$HOME/.profile`, the value applies to all X clients you run. Non-clients will not find their resource files unless you link or copy them into the directory specified by `XUSERFILESEARCHPATH`.

### Setting the KBD_LANG Environment Variable

X allows you to override the physical keyboard attaches to the HP-HIL.

This variable can be set after the server has started. The NLIO processes for Asian users start only when either the physical keyboard is Asian or KBD_LANG is set to an Asian language.

It should be necessary to set this variable only for those languages whose characters cannot be generated from the keyboard. For example, all HP roman8 or iso8859.1 characters can be generated by any western European keyboard. But to generate Japanese characters from a U.S. keyboard, you must set KBD_LANG to japanese.

### Language-Dependent Bitmaps—the XBMLANGPATH Variable

The XBMLANGPATH variable specifies the search path for language-dependent bitmaps. It lists the paths for bitmaps in this order:

1. User-specific bitmaps.
2. System bitmaps listed in the XmGetPixmap(3x) man page.
3. Append:

   ```
   /usr/lib/X11/bitmaps/%N/%B
   ```

   This ensures that you will get the non-localized bitmaps, where necessary.

## Other Language-Dependent Resource Files

When LANG is set, X uses the following language-dependent default resource file:

```
/usr/lib/X11/%L/sys.mwmrc
```

## Native Language Fonts

For information about using non-English fonts, refer to "Using Native Language Input/Output" in chapter 6.

# Using the X Window System

This chapter covers the basics of window operation. It shows you how to use X once it's been installed on your system. You'll learn how to perform the following tasks:

- Start the X Window System.
- Create, move, resize, and "shuffle" windows.
- Iconify a window and normalize an icon.
- Display menus and make selections.
- Stop programs and correctly exit your X environment.

The following chapters contain related information:

- Chapter 3 explains configuration files used by the X Window System.
- Chapter 7 explains the window manager (`mwm`) in more detail.

## Starting the X Window System

Before you start the X Window System, you must be logged in to your computer system. Log in using your normal procedure.

You should start the X Window System just once. With X11 running, you should *not* execute the `x11start` command again. Starting X11 and then starting it again while it is still running may cause undesirable results.

Note, however, that you can restart the *window manager* and refresh the *screen* at any time.

4 X will use the default `.x11start`, `.Xdefaults`, and `.mwmrc` files, unless told otherwise in the command line options.

### Starting X at Login

Your system may be configured to start X11 as part of the login procedure. If so, skip the rest of this section and the next and start reading at "What to Expect When X Starts."

### Starting X from the Command Line

If your system is not configured to start X11 at login, log into the system in the usual way and type the following command at the command prompt:

```
x11start (Return)
```

### Command-Line Options for x11start

In most cases, you will find it convenient to establish environment options in configuration files in your home directory. However, if you don't start X11 automatically at login, you can include environment options on the command line after the `x11start` command. The syntax for this is:

`x11start` [*-clientoptions*]`--` [*{path}/server*] [*:display*] [*-options*]

*Client options* pass from the `x11start` command line to all clients in the `.x11start` file that have a `$@` parameter. The options replace the parameter. This method is most often used to specify a display other than the usual one on which to display the client. You can, however, use the command-line option to specify a non-default parameter, such as a different background color.

The default `.x11start` file starts the following clients:

- A terminal emulation client, such as `hpterm`.
- `mwm`.

*Server options* are preceded with a double hyphen (--). If the option following the double hyphen begins with a slash (/) or a path and a slash, it starts a server other than the default server. If the option begins with a colon followed by a digit (:#), it specifies the display number (0 is the default display number). Additional options specified after the server or display refer to the specified server or display. See the `XSERVER` page in the reference section for more information on server options.

The examples below illustrate starting the X Window System in different ways.

| | |
|---|---|
| `x11start` | *The usual way to start X.* |
| `x11start -bg Blue` | *Gives clients followed by* `$@` *a blue background.* |
| `x11start -- /X2` | *Starts server X2 rather than the default server.* |

## Starting X on a HP-UX Multi-Display System

A multi-seat system (a system with more than one display, keyboard, and mouse) requires modification of two X11 configuration files, to allow for more than one display seat. These files, `X*screens` and `X*devices` (where `*` is the number of the display), are located in your system directory. Each seat must have its own `X*screens` and `X*devices` files. If you have a multi-seat system but have not configured it, see your system installation or configuration manual for more information. Also see "Defining Your Display" in chapter 7.

Seat 0 uses the `X0screens` and `X0devices` files to configure its output and input devices. These files are supplied with the system, but you must still match them to your hardware configuration. To start X11 on seat 0 (display 0) of a multi-seat system, log in as usual and type:

```
x11start (Return)
```

To start X11 on seat 1 (display 1) of a multi-seat system, log in as usual and type:

```
x11start -- :1 (Return)
```

Here the -- signifies starting the default server while the `:1` specifies sending the output to seat 1. Seat 1 uses the `X1screens` and `X1devices` files to configure its output and input devices. If your system has a multi-seat configuration, you must create these configuration files using the `X0screens` and `X0devices` files as models.

## What to Expect When X Starts

Whether you start the X Window System from the command line or automatically from a login file, `x11start` always executes the same sequence of steps.

1. If necessary, it adds the your system directory to your `PATH` variable.
2. It looks in your home directory for a `.x11start` command file to read. If it doesn't find one, it reads `sys.x11start` in your system directory instead.
3. It starts `xinit`, which starts the server and any clients specified in the `.x11start` command file.
4. It looks in your home directory for a `.Xdefaults` configuration file to read. If it doesn't find one, it reads `sys.Xdefaults` in your system directory instead.
5. It reads the configuration file named by the `$ENVIRONMENT` variable, `.Xdefaults-`*hostname* if the variable doesn't exist.

You won't notice any effect from issuing the command until the X display server starts.

### The Server Creates the Root Window

When `x11start` starts the server (the program that controls the operation of your keyboard, mouse, and display), your screen will turn gray. This means that the screen has now become the **root window**, the backdrop or "desktop" on which the windows and icons of your environment appear. Although you can completely cover the root window with clients, you can never cover a client with the root window. The root window is *always* the backdrop of your window environment; nothing gets behind it.

In the center of the root window is an hourglass. This is the **pointer** and marks the current screen location of the mouse.

### A Terminal Window Appears on the Root Window

A short time later the pointer changes to an ×, and a terminal window appears at the top of your display (if you're using the default `.x11start` file). This window is under the control of a window manager. If you use the OSF/Motif Window Manager (`mwm`), your window has a three-dimensional frame. This frame contains window manager controls.

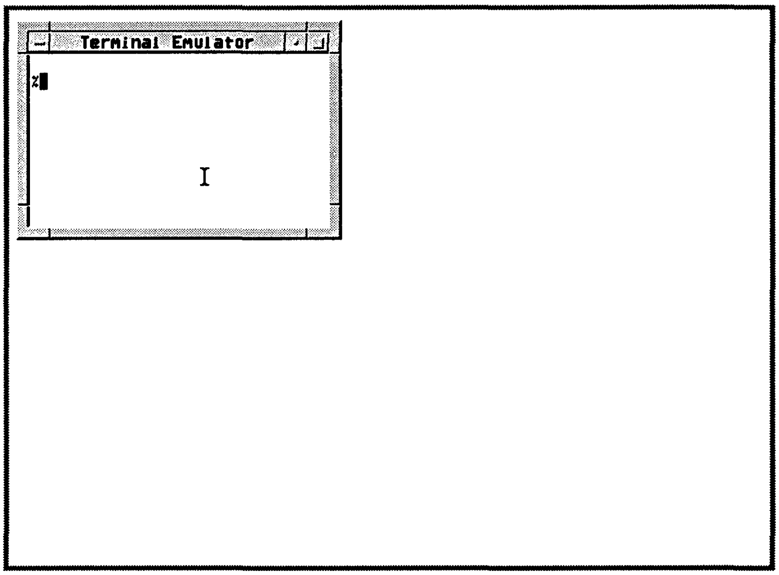


**Figure 4-1. The Default X Environment: 'mwm' and One Window**

The window contains a command-line prompt and behaves exactly like the screen of a terminal. You can think of this window as "a terminal in a window." There are two terminal emulation clients: `hpterm` and `xterm`. The examples in this book use `hpterm`. Refer to the man page for your terminal emulator for specific details about it.

Move the mouse. The pointer moves on the screen. When the pointer is in the root window, it has an × shape. However, when you move the pointer to a terminal window, the pointer changes to an arrowhead (when on the window frame) or an I (when in the interior of the window).

With the OSF/Motif Window Manager, when you press and release button 1 while the pointer is in a terminal window, the window becomes the **active window**. When a window is active, its frame changes color. You'll discover that you can't type in a terminal window unless the window is active.

The active window is the terminal window where what you type on the keyboard appears. *Your input always goes to the active window.*

If there is no active window, what you type is lost.

The program running in the active window decides what to do with your typed input. Frequently the program will use a **text cursor** to show where your typed input will be displayed.

## What to Do If X11 Doesn't Start

**Table 4-1. Possible X Window System Start Problems**

| If this happens ... | You should do this ... |
|---|---|
| The message `command not found` appears. | Check your spelling and reenter the start command. |
| The root window displays for a moment, but then goes blank. | Press the (Return) key to bring back your original command-line prompt and see below. |
| The root window displays, but no pointer appears. | Press (CTRL) (Left Shift) (Reset) all at the same time. This brings your original command-line prompt back. See below. |
| The root window and pointer display, but no terminal window appears. | Press *and hold* button 3. If a menu appears, open a window. Otherwise, press (CTRL) (Shift) (Reset) Try restarting X, then see below if there's still a problem. |
| The terminal window displays, but what you type doesn't appear after the window's command prompt. | Move the pointer into the window and click (press and release) button 1, then type. |

If you encounter problems starting X11 for the first time, check the following areas:

- Check the X11 start log in your home directory for clues by typing

  ```
  more .x11startlog (Return)
  ```

- Check that the correct directory is in your PATH statement. If you do not have an entry for the system directory, `x11start` will add that entry to the path. You can be sure that the entry is always there by adding it to the path yourself. To check the PATH variable, type

  ```
  env (Return)
  ```

- Check that the `DISPLAY` environment variable is set correctly. If you do not already have an entry for either `local:0.0` or *host:0.0* (where *host* is the hostname of your system), X11 will add it for you when X11 starts. You can add the entry yourself. To check the `DISPLAY` environment variable, type:

  ```
  env (Return)
  ```

- Check that you have the correct permissions for the `.x11start` file in your home directory. Type:

  ```
  ll .x11start (Return)
  ```

  The resulting permission should be at least:

  ```
  -rwx------
  ```

- Check the `.x11start` file in your home directory for errors. Compare it with the `sys.x11start` file in your system directory.

If none of the above seems to help, or you're not sure how to proceed, see your system administrator.

## Working With Windows

This section explains features of the OSF/Motif Window Manager (`mwm`).

In the typical X environment, you have two tools to control window operations:

- The mouse.
- The window manager.

For most window operations, you'll use a combination of the window manager and mouse. (If you lack the space on your desktop, or feel more comfortable with a keyboard, you can configure your keyboard to take the place of the mouse.)

### Using the Mouse

The X Window System works with either a two-button mouse or a three-button mouse.

**Table 4-2. Which Mouse Button Is Which**

| **To press this ...** | **On a 2-button mouse press ...** | **On a 3-button mouse press ...** |
|---|---|---|
| Button 1 | the left button | the left button |
| Button 2 | both buttons | the middle button |
| Button 3 | the right button | the right button |

Besides using the mouse to point with, you use the mouse buttons to select an operation to be performed on the object pointed to. Buttons have the following actions associated with them:

| | |
|---|---|
| Press | To hold down a button. |
| Click | To press *and release* a button without moving the pointer. |
| Double-click | To click a button twice in rapid succession. |
| Drag | To press *and hold down* a button while moving the pointer. |

## The Anatomy of an mwm Window Frame

mwm surrounds each window on the root window with a functional frame. Positioning the pointer on a part of the frame and performing a mouse button action will execute the function of that part of the frame.

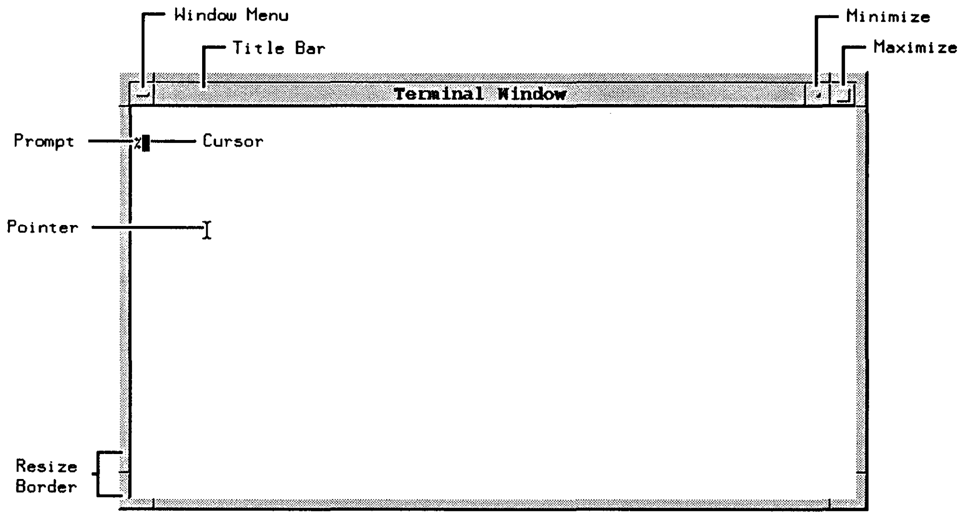


**Figure 4-2. Window Frame Parts**

The parts of the mwm window manager, their functions, and the required mouse operations are listed in the following table.

Table 4-3. Window Frame Parts and What They Do

| Frame Part | Function | Mouse Action |
|---|---|---|
| Title area | Move a window. | Press and drag button 1. |
| Window menu button | Display a window menu. | Press button 1. |
| Window menu button | Select a window menu item. | Press and drag button 1. |
| Window menu button | Close a window. | Double click button 1. |
| Minimize button | Iconify a window. | Press button 1. |
| Maximize button | Expand window to maximum size. | Press button 1. |
| Frame border | Stretch or shrink a window horizontally, vertically, or diagonally (in two directions). | Press and drag button 1. |
| Frame and window | Keyboard focus selection. | Press button 1. |
| Frame and window | (On focus selection) Top window. | Press button 1. |

## Activating a Window

You make a window active by moving the pointer to any part of the window and clicking button 1 of the mouse. When a window is active, you can interact with it.

## Changing Window Size and Location

You can change the size of a window, change its location on the screen, and change its location on the stack

## Moving a Window around the Screen

You can move any window (except the root window) by doing the following:

1. Position the mouse pointer in the title bar.
2. Grab the title bar by pressing *and holding down* button 1.
3. Drag the pointer. An outline of the window shows you the window's new location.
4. Position the outline and release button 1 to relocate the window.

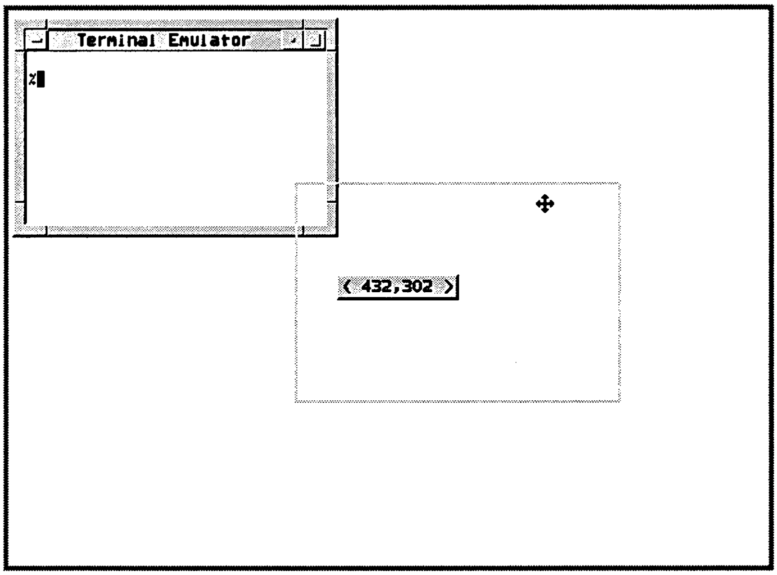


**Figure 4-3. An Outline Shows the Window's Location**

You will notice that, along with the window outline, a small location box displays at the center of the screen. The numbers in this box are the column and row position of the upper left corner of the actual window (the area inside the window frame). The measurement is in **pixels**. Pixels (short for picture elements) are tiny dots, arranged in rows and columns on the screen, that make up the displayed images.

As mentioned in the previous section, you can also move a window by choosing the "Move" selection from the window menu.

## Changing the Size of a Window

To change the size of a window, grab the window's frame with the pointer, drag the frame to the desired size, and then release the frame.

Where you grab the frame determines how the window gets resized. If you grab the side of the frame, the window stretches or shrinks horizontally. If you grab the top or bottom of the frame, the window stretches or shrinks vertically. If you grab the frame by one of the corner pieces, you can expand or contract the size of the window in two directions at once.

**Table 4-4. Where to Grab a Window Frame**

| **If you want to stretch or shrink the window . . .** | **Position the pointer on the . . .** |
|---|---|
| vertically from the . . . | |
| top | top of the frame, above the title bar |
| bottom | bottom of the frame |
| horizontally from the . . . | |
| right | right side of the frame |
| left | left side of the frame |
| diagonally from the . . . | |
| bottom left corner | frame's lower left corner |
| top left | frame's upper left corner |
| top right | frame's upper right corner |
| bottom right | frame's lower right corner |

The pointer changes shape when you're positioned correctly for the grab.

Follow these steps to grab and resize the window:

1. Position the mouse pointer on a part of the window frame.
2. Press *and hold* button 1.
3. Drag the mouse pointer. An elastic outline represents the new window size.
4. Release button 1 when the elastic outline is the correct size.

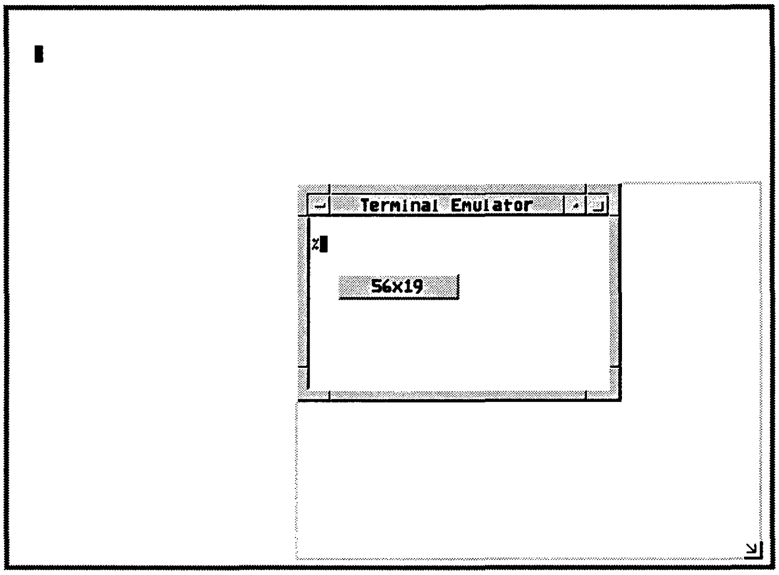


**Figure 4-4. An Elastic Outline Shows the Window Size**

Although you change a window's size and shape during a resize operation, you do not change its position. The section of the frame opposite where you grab always remains in the same location.

As mentioned earlier, you can also resize a window by choosing the "Size" selection from the window menu. If you choose the "Size" selection, you must cross the window frame's border with the pointer before the elastic outline appears.

## Raising a Window to the Top of the Window Stack

To raise a window to the top of the stack (front of the screen), position the pointer on any visible piece of the obscured window and click button 1. This also makes the window the active window.

An alternative in some situations is to lower the window on top of the stack by choosing the "Lower" selection from that window menu.

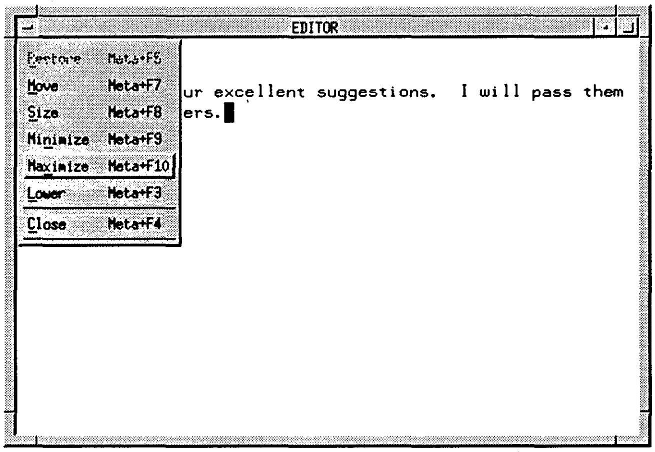
## Using Menus

Menus provide a quick way of performing some action. You make a selection from a list and the action is performed for you.

There are three types of menus you can use:

- window
- icon
- root

### Selecting from the Window Menu



ur excellent suggestions. I will pass them
ers.

**Figure 4-5. Every Window Has a Window Menu**

Every window has a window menu. The window menu button of a window is in the upper left corner of the window frame next to the title bar. You can display the window menu at any time by pressing button 1 with the mouse pointer on the window menu button.

There are three ways to display and use window menus.

- A **sticky menu** stays displayed until you make a choice. To display the window menu as a sticky menu:

  1. Position the pointer on the window menu button.
  2. Click button 1.

3. Move the pointer to the selection you want to choose.
4. Click button 1 on that selection. The window menu will disappear and the desired action will take place.

- A **pulldown menu** is displayed as long as a button is pressed. To display a window's window menu and make a selection:

  1. Position the pointer on the window menu button.
  2. Press *and hold down* button 1.
  3. Drag the pointer down the menu to the selection you want to choose.
  4. When the selection highlights, release button 1.
  5. (Move and Size only.) Move the pointer to the desired location or until the desired position or size is achieved, then click button 1 to end the operation.

  If you change your mind and don't want to make a selection, move the pointer off the menu area *before* you release button 1.

- You can also display the window menu by pressing (Left Shift) (ESC). To make a choice using this method, use the (▲) and (▼) keys to highlight a selection, then press (Return). If you don't want to make a selection, press (Left Shift) (ESC) again.

## Window Menu Selections

The following table describes the window menu selections.

**Table 4-5. The Window Menu Selections**

| To do this ... | Select ... |
|---|---|
| Restore a window from an icon or after maximizing. | Restore |
| Change the location of a window. | Move |
| Change the width and height of a window. | Size |
| Shrink a window to its icon (graphic representation). | Minimize |
| Enlarge a window to cover the entire root window. | Maximize |
| Send a window to the back or bottom of the window stack, the position closest to the root window. | Lower |
| Immediately stop the window and make it disappear. | Close |

You can also use mnemonics and accelerators to select items from the window menu. An accelerator is a key that selects a menu item without posting the menu. For example, the accelerator Alt F9 minimizes a window. The accelerators are shown on the right side of the menu items.

Mnemonics let you select a menu item once the menu has been posted. The mnemonic for a menu item is indicated by an underlined character in its label. To select a menu item using its mnemonic, press the unshifted key for the underlined character.

The rest of this chapter explains how you can use the mouse and the window manager to control the windows in your environment.

## Displaying and Selecting from the Root Menu

The root window has its own menu called (not surprisingly) the **root menu**. You can display the root menu any time the mouse pointer is on the root window. When the pointer is in the root window, remember, it has an × shape.

To display and select from the root menu:

1. Position the pointer anywhere in the root window.
2. Press *and hold* button 3 to display the menu.
3. Drag the pointer down the menu until you have highlighted the desired selection.
4. Release button 3.

To make no selection, move the pointer off the menu *before* you release button 3.

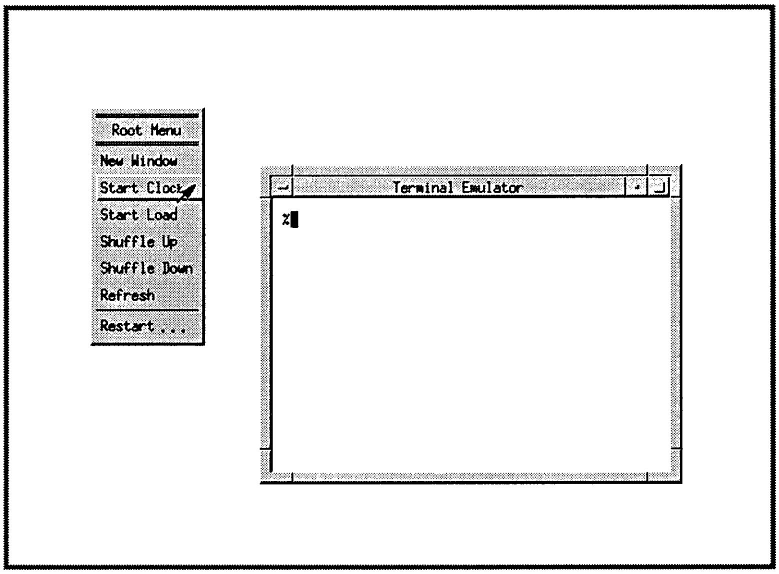


**Figure 4-6. The Root Menu Provides Screen-Wide Functions**

The default selections of the root menu provide you with screen-wide functions not appropriate for an individual window's window menu.

**Table 4-6. What the Root Menu Default Selections Do**

| To do this ... | Choose this selection ... |
|---|---|
| Make a new 80×24 `hpterm` terminal window near center screen. | New Window |
| Display an analog clock in the upper right corner of the root window. | Start Clock |
| Display a histogram measuring system load (displays next to the clock). | Start Load |
| Bring the most concealed window to the front of the window stack. | Shuffle Up |
| Lower the least concealed window to the bottom of the window stack. | Shuffle Down |
| Blank out then redisplay the screen (useful if video images become corrupt). | Refresh |
| Restart window manager to see recent configuration changes. | Restart |

## Icons

You can save space and bring order to your workspace by reducing inactive windows to icons—small, easily-recognizable graphic images that represent full-sized windows. Later, as you need them, you can change the icons back into full-sized windows.

Although you can't enter information into an icon, any program running in a window as it is iconified continues uninterrupted until it either completes or pauses to await input from you.

Icons allow you to start an application in a window and then collapse the window into a tiny symbol over in the corner of your screen. There the program quietly does its work without cluttering up your workspace.

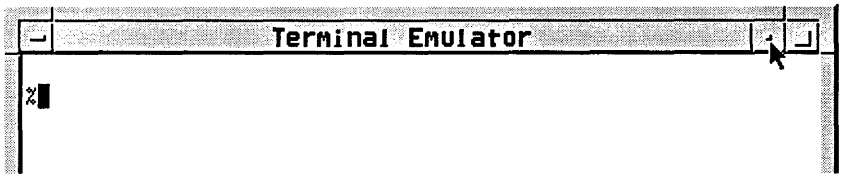


**Figure 4-7. Pressing the Minimize Button Iconifies a Window**

Changing a window into an icon is known as **iconifying** or **minimizing** the window. To iconify a window:

1. Move the pointer to the minimize button located in the upper right corner of the window frame between the title bar and the Maximize button.
2. Press *and release* button 1.

Immediately after you release button 1, the window is iconified. Successive icons are placed from left to right in a row along the bottom of the root window using a grid pattern. This placement is by default and can be changed if your needs require it.

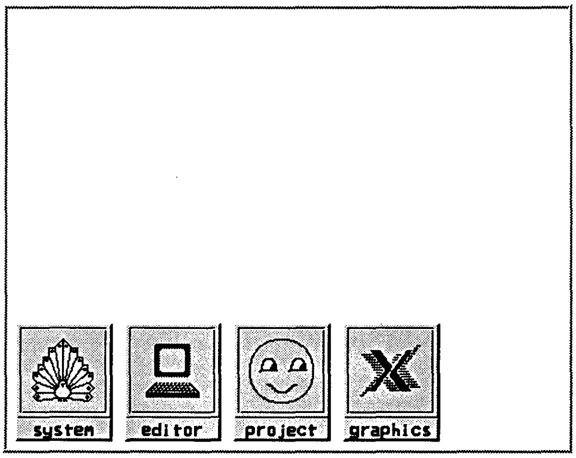


**Figure 4-8. Default Icon Placement Is along the Screen's Bottom**

You can also change a window into an icon by choosing the "Minimize" selection of the window menu as discussed earlier.

## Turning an Icon Back into a Window

When you have room on the root window, or simply want to check the progress of an application running in an iconified window, you can turn the icon back into a window. Changing an icon into a window is called **normalizing** or **restoring**.

1. Move the pointer to the icon.
2. Double-click button 1 (press *and release* it twice in rapid succession).

After you double-click on the icon, the window will reappear located at its previous (pre-iconified) position.

## Displaying and Selecting from an Icon's Menu

Although an icon doesn't have a frame like a window, it does have a window menu that gives you most standard control options. "Size" and "Minimize" appear on the menu but don't function with iconified windows.

To display an icon's window menu and make a selection:

1. Move the mouse pointer over the icon.
2. Click button 1 to activate the icon and display the menu.
3. Move the mouse pointer to the selection you want.
4. Click button one to make the selection.

To make no selection, move the pointer to the root window and click button 1. The icon will stay active until you make another window or icon active.

## Moving Icons around the Screen

Although icons appear by default in a row along the bottom of the screen, you can move them anywhere on the root window.

To move an icon:

1. Move the mouse pointer onto the icon.
2. Press *and hold* button 1.
3. Drag the pointer to a new location. An outline of the icon shows the current location.
4. Release button 1.

## Exiting From the X Window System

Exiting from the X Window System means stopping the X11 display server. Leaving X places you back at the command prompt you had immediately after you logged into your system.

Before stopping the X Window System, you must first stop any X clients you may have running. This ensures that you do not unknowingly leave any orphaned processes executing. It also ensures that all open files are properly closed to prevent loss of data.

**Caution** Stop all X clients and any non-clients running in terminal emulator windows before stopping the window system. If you don't do this, any open files may not be updated properly. This could result in the loss of valuable data.

### Stopping Application Programs

You can stop a program and remove its window in three ways.

#### Following the Program's Normal Exit Procedure

The best way to exit a program is to use the program's usual "exit" procedure. This should always be your preferred method for stopping the program. Many programs have commands or keystrokes that stop them.

If the program is a client and created its own window, the window is removed when the client stops. If the program is a non-client in a terminal window, the window remains, and you can stop it when you stop the display server.

## Closing the Window

You can also stop most applications by closing the window in which the application is running. To close a window:

1. Position the pointer on the window menu button.
2. Press *and hold* button 1.
3. Drag the pointer to Close.
4. Release button 1.

## Stopping the X Window System

After stopping all application programs, stop the window system by holding down the (CTRL) and (Left Shift) keys, and then pressing the (Reset) key. This stops the display server, and with it the window system.

The sequence of keys that stops the display server can be customized in the `X*pointerkeys` file. Refer to chapter 9 or the `X0pointerkeys` file in your system directory.

5

# Application Resources

Resources are data used by applications to set their appearance and behavior.

This chapter covers:

- The various ways to change resource settings.
- The scope of resources— how specifically or generally a resource is applied.
- The syntax for specifying color and geometry resources.

## How Applications Obtain Attributes

An application can get attributes from several different places:

- Resources directly loaded into an application's resource database:
  - Command-line options.
  - `.Xdefaults` file.
  - Resources loaded into the RESOURCE_MANAGER property.
  - Application resource files (for example, `app-defaults` files or `.rc` files).
- Other sources:
  - Defaults built into the client.
  - Environment variables.
  - Inter-client communications.

The following list shows how applications obtain resources. A resource at the top of the list overrides the same resource found further down the list. For instance, a resource in `.Xdefaults` overrides the same resource in the `app-defaults` directory.

- Command-line options. These options are good for only that one instance of the application. A command-line option is the equivalent of a *client.resource* statement in a resource file.
- A host environment:
  - If an `XENVIRONMENT` variable exists, it may contain the name of a file that specifies application attributes.
  - A `$HOME/.Xdefaults-`*host* file may contain resources to be used for a specific remote host. It is read only if no `XENVIRONMENT` variable exists.
- Personal resources:
  - Loaded into the RESOURCE_MANAGER property.
  - `.Xdefaults` (or `sys.Xdefaults`) file.
- User-specific files for particular classes of applications:
  - If an `XUSERFILESEARCHPATH` variable exists, it may specify a directory of files containing application class defaults for the specific user.
  - If `XUSERFILESEARCHPATH` variable does not exist, and if an `XAPPLRESDIR` variable exists, it may specify a directory of files containing user-specific application class defaults.
  - `$HOME/`*app-class* files may contain application resources. These files are read only if `XUSERFILESEARCHPATH` and `XAPPLRESDIR` do not exist.

  For information about these variables, refer to *Programming with the Xt Intrinsics*.
- Application-specific configuration files in the `app-defaults` subdirectory of the system directory. Each file specifies attributes for a particular class of application. An `app-defaults` file is the equivalent of a *Class*resource* statement in a resource file. (The environment variable `XFILESEARCHPATH` may define a language-dependent location of `app-defaults`.)
- Internal defaults built into the application.

## Ways to Change Resources

There are several ways to change a resource. The way you choose depends on:

- The nature of the resource.
- When you want the change to take effect—immediately or at the beginning of the next session.

Resources can be changed by:

- Loading the new resources into the server's RESOURCE_MANAGER property using the X client `xrdb`.
- Hand editing a resource file, such as `.Xdefaults`.
- Using command-line options.

## Setting Resources with .Xdefaults

The `.Xdefaults` file contains default resources you want to apply each time a client is started.

If you do nothing, the system uses the the `sys.Xdefaults` file in your system directory. If you want your own defaults to be used instead, copy this file into `.Xdefaults` in your home directory and make modifications there. For example:

```
cp /usr/lib/X11/sys.Xdefaults $HOME/.Xdefaults
```

The syntax for describing resources is explained later in this chapter.

## Changing the RESOURCE_MANAGER Property with 'xrdb'

The RESOURCE_MANAGER property is a property on the root window that is treated the same way as a resource file by the resource manager.

During a session, the RESOURCE_MANAGER property may be modified by the the `xrdb` client.

You can use `xrdb` to load resources into the server's RESOURCE_MANAGER property.

The syntax for `xrdb` is:

```
      ┌ -help                              ┐
      │ { -cpp   } path/filename           │
      │ { -nocpp }                         │
      │ -symbols                           │
      │ -query                             │
      │ ⎧ -load path/filename  ⎫           │
xrdb  │ ⎨ -merge path/filename ⎬           │  [filename]
      │ ⎪ -remove              ⎪           │
      │ ⎩ -edit path/filename  ⎭           │
      │ -backup string                     │
      │ -Dname[=value]                     │
      │ -Uname                             │
      │ -Ipath/directory                   │
      └ -display host:display              ┘
```

| | |
|---|---|
| `-help` | Displays a list of options for `xrdb`. |
| `-display` | Specifies the host and display of the server to be loaded with the configuration information. |
| `-query` | Displays the current contents of the server's RESOURCE_MANAGER property. |
| `-load` | Specifies that `xrdb` should load the file named on the command line into the RESOURCE_MANAGER property, overwriting the current resources listed there. This is the default action. |

| | |
|---|---|
| `-merge` | Specifies that `xrdb` should load the file named on the command line into the RESOURCE_MANAGER property, merging the new resources with the current resources instead of overwriting them. |
| `-remove` | Removes the current configuration file from the RESOURCE_MANAGER property. |
| `-edit` | Places the contents of the RESOURCE_MANAGER property into the named file, overwriting resources specified there. |
| `-backup` | Specifies a suffix to be appended to the filename used in the `-edit` option to create a backup file. |
| `-cpp` | Specifies the path and filename of the C preprocessor to use when loading a configuration file containing `#ifdef` or `#include` statements. `xrdb` works with CPP and other preprocessors as long as they accept the `-D`, `-U`, and `-I` options. |
| `-nocpp` | Specifies that `xrdb` should not use a preprocessor before loading the configuration file (the file contains no statements that need preprocessing). |
| `-symbols` | Displays the symbols currently defined for the preprocessor. |
| `-D`*name* | Defines a symbol for use with conditional statements in the configuration file used by the RESOURCE_MANAGER property. |
| `-U`*name* | Removes a defined symbol from the RESOURCE_MANAGER property. |
| `-I`*path/directory* | Specifies the search path and directory of `#include` files used in the RESOURCE_MANAGER. |

To add resources interactively:

1. Execute:

   ```
   xrdb -merge -nocpp
   ```

   in a local terminal emulation window.

2. Type in the resource specifications. Each resource must be on a separate line.

3. When you've typed all the resources, press CTRL d to merge the resources and restore the shell prompt.

To add resources by typing the resources into a file that is then merged into the database:

1. Create a file containing the resources you want to add.
2. Execute:

   ```
   xrdb -merge -nocpp filename
   ```

## Syntax of Resource Specifications

Resource files are text files. They must obey the following syntax rules:

- Each resource specification must be on a separate line. If the last character on a line is a backslash (\), the new-line following the backslash is ignored and the resource specification is assumed to continue on the next line.
- To add comments to resource files:
  - Use the exclamation (`!`) character. Anything to the right of the `!` is interpreted as a comment. This is the preferred way of commenting all or portions of lines.
  - You can place a pound (`#`) character in column 1. This makes the entire line a comment. Keep in mind that you must use the `xrdb` option `-nocpp` when loading a commented resource to avoid it being interpreted as a preprocessor directive.
- The resource name is separated from the value by a colon (`:`) and optional spaces or tabs.

The general syntax for specifying a resource for a client is:

$$\left[\left\{\begin{array}{l}\textit{client\_name}\\ \textit{client\_class}\end{array}\right\}\right] * \ \textit{resource}\texttt{:}\ \textit{value}$$

For example:

```
hpterm*background: skyblue
```

sets the background color of the `hpterm` window to skyblue.

Trailing blanks in a resource value are parsed and therefore can cause errors to occur. For example, if you inadvertently included a blank after "skyblue" in the example above, several warning messages appear when the program using the resource definition is run:

```
Warning: Color name "skyblue " is not defined
Warning: Cannot parse default background color spec
```

Certain clients allow you to set resources for particular parts of the client. For example,

5

```
hpterm*scrollBar*background: mediumblue
```

sets the scrollbar on `hpterm` windows to mediumblue.

## Scope of Resource

You can specify how generally or specifically a resource is applied. For example, you can specify that all clients have a background color of black (very general). At the other extreme, you can say that you want the softkeys of one particular `hpterm` window to be red.

Scope of customization is determined by:

- Using names or classes of clients.
- Using names or classes of resources.
- Specifying particular areas of clients (for example, softkeys and scrollbars).
- Using wildcards in the resource string.

## Names and Classes of Clients

Every client has both a name and a class. The name defines the specific client, while the class categorizes the client. Thus, the class is more general than the name.

Frequently, the two identifiers are very similar, and often differ only in capitalization. For example, the client named xclock belongs to class Xclock.

Resources specified by client name take precedence over resources specified by client class.

## Naming a Client

You can assign a name to a particular instance of a client. This allows you to allocate resources to that client by class, by client, *and* by name.

For example, the following command line starts an instance of hpterm named localTerminal.

```
hpterm -name localTerminal
```

If the following resource exists in the resource database:

```
HPterm.name:                 localTerminal
localTerminal*background     white
```

then the localTerminal window will be white, overriding the colors used by the current palette.

## Names and Classes of Resources

Like clients, resources have both a name and a class.

An individual resource begins with a lowercase letter. For example, foreground refers to the foreground resource. A class resource, however, begins with an upper-case letter. For example, Foreground refers to the entire class of foreground resources.

Thus, if no other specifications overruled, the line *foreground: blue in your resource file would make all foregrounds blue. However, the line *Foreground: blue would make all resources that belonged to the Foreground class blue. This would include such resources as foreground, cursorColor, pointerColor, bottomShadowColor for softkeys, frames, icons, and mattes.

## Name/Class Precedence

Specific resource specifications always have precedence over general specifications. For example, suppose a resource file contains:

```
*Foreground:           red
HPterm*Foreground:     DarkSlateGray
HPterm*foreground:     coral
HPterm*cursorColor:    green
```

The first line makes all resources of the class Foreground red. The second line overrules the first line, but *only* in the case of clients of class HPterm (of which there is only one–the hpterm client itself). Line two makes the Foreground class resources of all hpterm clients DarkSlateGray. Lines three and four give hpterm clients coral foregrounds and green cursors, while the other resources of class Foreground (pointerColor, cursorColor, softkey foreground and bottomShadowColor, and scrollbar foreground and bottomShadowColor) remain DarkSlateGray for hpterm clients.

Similarly, if a resource file contains:

```
hpterm.name:                 local
HPterm*softkey*background:   wheat
HPterm*background:           pink
local*background:            white
```

then all softkey backgrounds will be wheat. For the rest of the hpterm window, the backgrounds will vary. Windows named local will be white, other windows will be pink.

## Wildcards and Exact Paths

The * character in a resource string is a wildcard that provides resource generality. For example, the following list of resources shows increasing specificity.

```
*foreground:                 white
hpterm*foreground:           yellow
hpterm*softkey*foreground:   red
```

The resource *foreground refers to *all* foregrounds. The more specific resources override it. All the hpterm foregrounds will be yellow except for the foreground of the softkeys.

## Color Resources

Resources take color names or rgb numbers as their values:

- The `rgb.txt` file in your system directory lists all the named colors.
- The rgb numbers have the syntax:

  #*RedGreenBlue*

  where *Red*, *Green*, and *Blue* are hexadecimal numbers containing 1, 2, 3, or 4 digits that indicate the amount of that primary color used. There must be the same number of digits *for each* of the primary colors. Thus, valid color values consist of 3, 6, 9, or 12 hexadecimal digits. For example, black can be specified by any of these rgb values: `#fff`, `#ffffff`, `#fffffffff`, or `#ffffffffffff`.

For example, the following line specifies the background color of `hpterm` icons:

```
Mwm*hpterm*iconImageBackground:   DarkSlateGrey
```

5

Refer to the man page for a specific client to see which if there are special elements for that client that can be colored. For example, `xclock` allows you to color the hands and tic marks in addition to the background, foreground, and window frame colors:

```
XClock*hands:       Skyblue
```

Or `xterm` allows you to control colors within a scrollbar:

```
Xterm*scrollBar*foreground:     Plum
```

## Geometry Resources

The geometry of a window is its size and location  The syntax for geometry resources is:

$$\left\{\begin{array}{l} Width\times Height \\ \pm column\pm row \\ Width\times Height\pm column\pm row \end{array}\right\}$$

Use a lower-case `x` for the times sign.

|  |  |  |
|---|---|---|
| *Width* | The width in characters (for terminal windows) or pixels (for other clients). For widths in characters, the window size depends on the font size. | |
| *Height* | The height of the window in lines (for terminal windows) or pixels (for other clients). The height of a terminal window depends on the font. | |
| *column* | The column location of the window in pixels. | |
| | Plus (+) values | The location of the left side of the window relative to the left side of the workspace. |
| | Minus (−) values | The location of the right side of the window relative to the right side of the workspace. |
| *row* | The row location of the window given in pixels: | |
| | Plus (+) values | The location of the top of the window relative to the top of the workspace. |
| | Minus (−) values | The location of the bottom of the window relative to the bottom of the workspace. |

**Table 5-1.**
**Example Locations for an 80×24 Terminal Window.**

| To position a window here ... | Use this location ... |
|---|---|
| The upper left corner of the workspace. | +1+1 |
| The lower left corner of the workspace. | +1−1 |
| The upper right corner of the workspace. | −1+1 |
| The lower right corner of the workspace. | −1−1 |

For example, the following line specifies that all `hpterm` windows be created 80 characters wide and 24 characters high, and that they are initially placed in the upper right corner of the display.

```
hpterm*geometry:   80x24-1+1
```

## Font Resources

There are four general font resources that are commonly used.

**Table 5-2. General Font Resources**

| Resource | Description |
|---|---|
| `Font` | General user font |
| `FontList` | Displayed in system areas of clients created using the OSF/Motif toolkit. |
| `XmText*FontList`<br>`XmTextField*FontList` | Displayed in text entry boxes of clients created using the OSF/Motif toolkit. |

Use the following syntax to specify font resources:

$\{$ *client_class client_name* $\}$**fontresource*: *fontname*

where:

| | |
|---|---|
| *client_class* | The class of the client for which you specify the font. |
| *client_name* | The name of the client for which you specify the font. |
| *fontresource* | The name of the font resource. |
| *fontname* | The name, alias, or xlfd name of the font. Refer to chapter 8 for information about how to specify font names. |

For example,

```
hpterm*Font:        fontname
```

Font resources and names are covered in more detail in chapter 6.

# 6

# Using Fonts

This chapter covers:

- Displaying samples of bitmapped fonts and scalable typefaces.
- Setting font resources.
- Using the X11R5 font server.
- Understanding and using the XLFD (X Logical Font Description) font name for bitmapped fonts and scalable typefaces.
- Administering bitmapped fonts and scalable typefaces.
- Using fonts with Native Language Support.

Chapters containing related information are:

- Chapter 5 covers where and when to set resources.
- Chapter 7 covers running clients from the command line.

A **font** is a type style in which text characters are printed. The X Window System includes a variety of fonts.

**Bitmapped fonts** are made from a matrix of dots. The font is completely contained in one file. Many files are needed to have a complete range of sizes, slants, and weights. Bitmapped font files can be read by the X server or the font server.

**Scalable typefaces** are each defined by a file containing a mathematical outline used by the system to create a bitmapped font for a particular size, slant, or weight. Scalable typefaces are readable by the font server. An X server wishing to use them must obtain them from a font server. See the sections entitled "Scalable Typeface Administration" and "Scalable Typefaces File Structure" in this chapter for more information.

Hewlett-Packard's X11R5 release of the font server supports two scalable font technologies: Agfa Compugraphic's Intellifont and Adobe's Type 1. Scalable outlines bundled with your operating system are Agfa's "CG Times," "Univers," and "Courier," and Adobe's "Utopia" and "Courier."

In addition to the scalable font technology available with the X11R5 font server, both the X server and the font server are now capable of rescaling bitmapped fonts to any size. This is not a recommended method of creating new fonts from existing ones — the results are often unsightly or even unreadable — but it is occasionally useful. The discussion of scaled fonts below also applies to scaled bitmaps, except where indicated.

The Intellifont Scalable Typeface Library from Agfa Compugraphic includes more than 180 different designs. Forty-six individual volumes of 4 faces are available, as well as six application-specific collections for all office publishing needs. Please call Agfa Compugraphic directly at 1-800-873-FONT (3668) for a free brochure, or contact your local Hewlett-Packard Sales and/or Customer Service office for more information.

## Customizing the Font Search Path with 'xset'

The X server must know where to find the fonts you want to use. The *font path* is a list of font sources accessible to the X Window System. A font source can be either a directory containing bitmapped fonts, or a font server accepting connections at some TCP address.

The `xset` command allows you to tell the X server which font sources to use. You specify directories containing bitmapped fonts by the complete path name. You specify font servers by the string "tcp/<hostname>:portnumber." To examine the font path, type:

```
xset q Return
```

To add or remove sources from the path:

`xset` *options*

where the options are:

| | |
|---|---|
| `-fp` *source[,source...]* `fp-` *source[,source...]* | Remove the directories from the head (-fp) or tail (fp-) of the font source. |

| | |
|---|---|
| `+fp` *source[,source...]* `fp+` *source[,source...]* | Adds the sources to the head (+fp) or tail (fp+) of the font source. |
| `fp=` *source[,source...]* | Specifies the complete font source. |
| `fp default` | Resets the default font source |
| `fp rehash` | Causes the server to reread the font databases for all directories (but not font servers) in the font path. This should be done after making any changes to directories that are in the font path, especially if you run `mkfontdir`, or if you change `fonts.alias` in any of these directories. |

Here are some examples that show you various ways to use `xset`.

```
xset fp tcp/:7000
```

Tells the X server to get all fonts from a font server running at TCP address 7000 on the local computer.

```
xset fp+ tcp/fontmaster:7000
```

Tells the X server to add a font server running on host `fontmaster` at TCP address 7000, to the end of the font path.

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

Tells the X server to add the directory `/usr/lib/X11/fonts/misc`, which contains bitmapped fonts, to the beginning of the font path.

More information about `xset` is presented in chapter 8.

## Listing Available Fonts with ‘xlsfonts’

The `xlsfonts` client lists fonts available to you. It uses the `fonts.dir` and any `fonts.alias` files in the font search path to find the fonts. The XLFD name or the alias name is listed. Refer to “The X Logical Font Description (XLFD)” for information about the XLFD name, and to “The fonts.alias File” for information about alias names.

The `xlsfonts` client has the following syntax:

`xlsfonts` [*-options*]

where *options* are:

| | |
|---|---|
| `-display` *host:display* | The X server whose fonts you wish to list. The default is the requesting display. |
| `-l` | Generate a medium listing. |
| `-ll` | Generate a long listing. |
| `-lll` | Generate a very long listing, showing individual character metrics. |
| `-m` | Long listings should show minimum and maximum bounds of each font. |
| `-C` | Multiple column listings. Same as `-n 0`. |
| `-1` | Single column listings. Same as `-n 1`. |
| `-w` *width* | Width in characters of each column. Default is 79. |
| `-n` *columns* | Number of columns for listings. |
| `-u` | Output is unsorted. |
| `-o` | Use `OpenFont` and `QueryFont` rather than `ListFonts`. |
| `-fn` *pattern* | `xlsfonts` will find all fonts that match this pattern. Wild cards may be used. If this option is not included `xlsfonts` lists all available fonts. |

A listing looks like this:

```
-adobe-courier-bold-o-normal--10-100-75-75-m-60-hp-roman8
-adobe-courier-bold-o-normal--12-120-75-75-m-70-hp-roman8
:
courb10
```

```
    courb12
    ⋮
```

The first two lines show the fonts' XLFD names, and the second two lines show the file name aliases for those fonts.

If you have many fonts on your system, `xlsfonts` can produce a long list. If you want to check for a specific font, use the pattern matching capability of `xlsfonts`. Use wild cards to replace the parts you are not trying to match. For instance, to see what scalable typefaces you have, type:

```
    xlsfonts -fn "*-0-0-0-0-*" (Return)
```

## Using the X11R5 Font Server

The Networked font server provides font services to one or more X servers in a networked environment. It allows a font administrator to distribute fonts to all X servers from a central administration point. The font server also provides increased font capabilities over those built into the X server — unlike the X server, the font server understands scalable fonts as well as bitmapped fonts.

In an environment without a font server, the X server is able to load bitmapped fonts from directories in its font path:

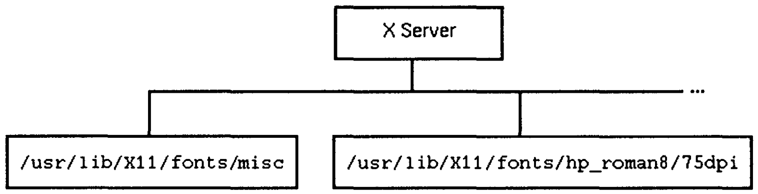


**Figure 6-1. An Environment Without a Font Server**

In an environment with a font server, the X server can also load fonts from a font server:

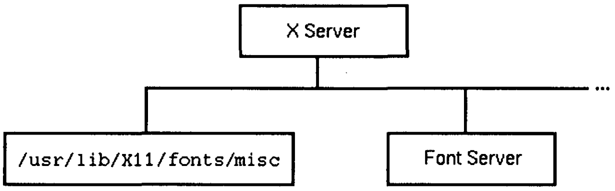


**Figure 6-2. An Environment With a Font Server**

The font server itself obtains fonts from directories or from other font servers. Unlike the X server, the font server can read scalable fonts as well as bitmapped fonts:

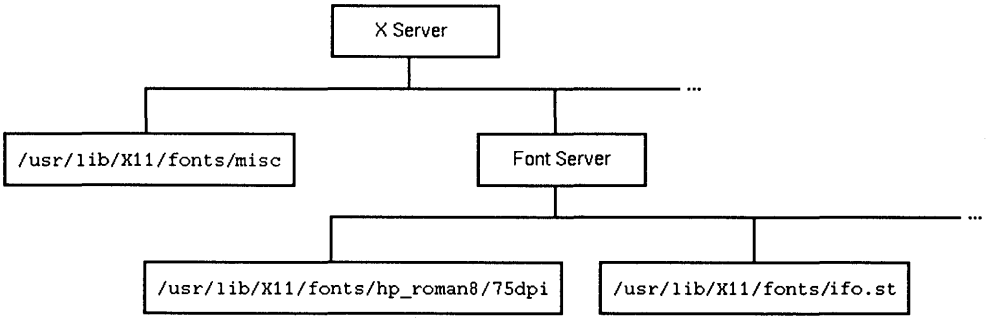


**Figure 6-3. Font Server Reads Scalable and Bitmapped Fonts**

The connection between the X server and a font server is over a TCP network connection: the font server can be running on the same machine as is the X server, or on a remote machine that is acting as a font source for multiple X servers.

A font server is shipped with Hewlett-Packard's X11R5 distribution but, by default, is not automatically started. To start the font server, execute:

```
/usr/bin/X11/fs -daemon
```

This starts a font server in its default configuration (see below for information on modifying font server configuration). Any X server started on the same system after the font server is started will automatically gain access to the fonts provided by that font server, including the licensed scalable Intellifont and Type 1 fonts in the `/usr/lib/X11/fonts/ifo.st` and `/usr/lib/X11/fonts/type1.st` directories.

## Managing the Font Server's Configuration

By default, a font server accepts connections from font clients (such as X servers) at TCP address 7000, and configures itself according to information in the file `/usr/lib/X11/fs/config`. Both of these defaults can be overridden with command-line options. See the fs(1) man page for more information.

Like the X server, a font server has a font path, a list of sources from which it can get fonts. There are three major differences between the font server's font path and the X server's font path:

- The font server font path is specified in the configuration file, and cannot be changed by a protocol request from a font client. For example, an X server can add or remove font servers to or from its font path, but it cannot tell a font server to change its own font path.
- The font server font path can be changed by modifying the configuration file and sending a signal SIGUSR1 to the font server.
- The font server font path can include directories containing scalable fonts, such as `/usr/lib/X11/fonts/ifo.st` and `/usr/lib/X11/fonts/type1.st`.

Like an X server, a font server can have font servers in its path. That is, font servers can be "chained." Font server chaining can be used to implement powerful and flexible networks of font sources. For example, figure 6-4 shows a company-wide font server with department-specific font servers.

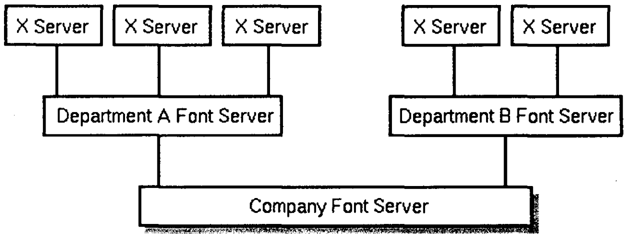


**Figure 6-4. Chaining Font Servers**

All users in Department A and Department B add their respective department's font server to their font path. The font administrator on each machine serving fonts then decides how to configure that font server for use by that department. In this case, he adds the company font server to the font path.

You can modify this path, as well as other configuration parameters, by editing the configuration file. See the fs(1) man page for information about all of the options in the configuration file.

### Starting the Font Server at Boot Time

Once you have determined your font server configuration — one font server per machine, a centralized font server, or whatever — and determined which machines you wish to use as font hosts, you should configure these hosts to start a server whenever they boot. You can do this by adding the appropriate commands to the `/etc/rc` startup file. For example, add the following definition to the function definitions at the beginning of the file:

```
start_fontserver()
{
 if [ -x /usr/bin/X11/fs ]
 then
  echo "starting fontserver"
  /usr/bin/X11/fs -daemon
 fi
}
```

You also need to add `start_fontserver` to the appropriate list of startup actions in the "Actions based on system type" section. As defined above, whenever the host boots, a font server starts at the default TCP address of 7000 using the default configuration file `/usr/lib/X11/fs/config`.

## The X Logical Font Description (XLFD)

The standard X interface provides a detailed description of the font to the X server by means of the *X logical font description* (XLFD) name. The XLFD name is a string of characters that describes properties of the font you want.

In X11R5, the XLFD standard supports both bitmapped and scalable fonts. In addition, HP has extended the standard to provide more capabilities with scalable fonts — that is, the ability to generate more variations on scalable fonts. These extensions are described in the following sections.

The form of the XLFD is 15 fields separated by dashes. These fields are explained later in this section.

"*FontNameRegistry-Foundry-FamilyName-WeightName-Slant -SetwidthName-AddStyleName-PixelSize-PointSize-ResolutionX -ResolutionY-Spacing-AverageWidth-CharSetRegistry -CharSetCoding*"

For example,

```
-adobe-courier-bold-o-normal--10-100-75-75-m-60-hp-roman8
```

specifies a courier, bold, oblique bitmapped font created by Adobe. The font is 10 pixels tall, 100 tenths of a point tall on a 75dpi×75dpi display. Characters are monospaced, and are an average of 60 tenths of a pixel wide. Fonts codes are based on the HP Roman8 encoding.

What is actually in the XLFD name differs depending on where in the font-request process the string is being used:

| | |
|---|---|
| reference XLFD | This is the XLFD name shown by `fonts.dir` and the `xlsfonts` client. Scalable typefaces have the PixelSize and PointSize fields set to zero. |
| request XLFD | This is the XLFD name you use to request a font. It is also the XLFD name you use in a `fonts.alias` file. Any field in the list can be replaced by the "`*`" wild card. Any character in the list can be replaced by the "`?`" wild card. |
| resolved XLFD | This is the XLFD name that the server returns when it has filled your font request. All the fields are filled in with the correct values for that font. However, they may not be the same values as in the request XLFD. |

## XLFD Syntax

This section explains the meaning of the fields in the XLFD name. Examples of the use of these fields are in a later section.

The XLFD name is long, so you can assign a shorter nickname, or alias, for the font, which you then use in place of the long string. Aliases are discussed in "The fonts.alias File" later in this chapter.

You may use either upper-case or lower-case letters when you specify a characteristic. Reference XLFD names are all lower-case.

The text "*[extensions]*" means that there are optional extensions to the standard XLFD fields that are used to generate additional font variations. Notice that the underscore (_) character is used in some extensions to avoid confusion with the dash (-).

## FontNameRegistry

The authority that registered the font. Usually left blank. If there is a value in the field, it is of the form *+version*, where *version* is the version of some future XLFD specification.

## Foundry

The name of the digital type foundry that digitized the font data.

## FamilyName

The trademarked commercial name of the font. If the FamilyName contains spaces, do one of the following for a request XLFD name:

- Enclose the entire XLFD name in double quotes ("). For example, this `fonts.alias` file line.

```
italic "-agfa-cg century schoolbook italic-normal-i-*---240---p-150-*-roman8"
```

6

- Use wild cards for part of the field.

```
italic -agfa-*schoolbook*italic-normal-i-*---240---p-150-*-roman8
```

## WeightName*[extensions]*

The relative weight of the font, such as bold.

For scalable typefaces, the user may specify that the font be darker (bolder) or lighter than the normal for that font. The syntax for this optional extension is:

$$\left[\begin{array}{l}\pm \mathit{horiz\_value} \\ \left[\pm \mathit{vert\_value}\right]\end{array}\right]$$

| | |
|---|---|
| *horiz_value*, *vert_value* | The increase (+) or decrease (_) in boldness. A value of 4000 for a normal font simulated the bold version of that font. |

If only one delta and value are specified, they apply to both directions.

ABCDEFGHIJKLMNOPQRSTUVWXYZ
ABCDEFGHIJKLMNOPQRSTUVWXYZ
**ABCDEFGHIJKLMNOPQRSTUVWXYZ**
**ABCDEFGHIJKLMNOPQRSTUVWXYZ**
**ABCDEFGHIJKLMNOPQRSTUVWXYZ**

**Figure 6-5. The Same Font at Increasing Weights**

### Slant*[extensions]*

A code indicating the direction of the slant for the font.

| | |
|---|---|
| `r` | Roman (no slant) |
| `i` | Italic (slant left) |
| `o` | Oblique (slant left) |
| `ri` | Reverse italic (slant right) |
| `ro` | Reverse oblique (slant right) |

For scalable typefaces, the user can request additional slanting from the normal. The syntax for this optional extension is:

*±value*

| | |
|---|---|
| *±value* | The angle in 1/64 degree ranging from 0° to 45° (0-2880). (0.5° = 32, 1° = 64, etc) Values outside of that range will return an error. Use + for counterclockwise angles, _ for clockwise angles. |

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
*A B C D E F G H I J K L M N O P Q R S T U V W X Y Z*
*A B C D E F G H I J K L M N O P Q R S T U V W X Y Z*
*A B C D E F G H I J K L M N O P Q R S T U V W X Y Z*

**Figure 6-6. The Same Font with Different Slants**

### SetwidthName

The width-per-unit of the font, such as compressed or expanded.

### **AddStyleName***[extensions]*

A description of anything else needed to uniquely identify the font, such as serif or cursive.

For scalable typefaces, users can specify that the font be mirrored or rotated. The syntax for the optional extension is:

$$\begin{bmatrix}\texttt{+Mx}\\ \texttt{+My}\end{bmatrix}\left[\pm \textit{angle}\right]$$

| | |
|---|---|
| `+Mx`, `+My` | `Mx` mirrors the font horizontally, and `My` mirrors the font vertically. |
| *angle* | ± the amount of rotation from normal in 1/64th degree increments. Use + for counterclockwise angles; use _ for clockwise angles. |

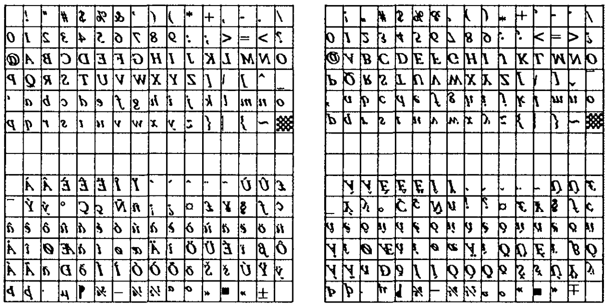


**Figure 6-7. Font Mirrored Horizontally and Vertically**

Don't confuse "slant" with "rotation". A character that has been slanted has its base in the normal position and the top pushed to one side. A character that has been rotated has been moved around some central pivot point.

### **PixelSize** *[Extensions}*

An integer describing the height of an EM square in pixels.

For scalable typefaces, you can increase or decrease the horizontal size to make a font wider or narrower than normal for that font. The syntax for this optional extension is

$$\left[\ +\textit{pixelwidth}\right]$$

*pixelwidth* The horizontal size in pixels. If this field is not specified, it is assumed to be the same as PixelSize.

For example, 20+10 requests a font 20 pixels high and 10 pixels wide (or, more accurately, a 20-pixel font whose width is half its normal width).

The PixelSize and PointSize fields are related through the ResolutionY field in the XLFD name (see below). You should specify a font by using either PixelSize or PointSize, but not both. An error occurs if you specify both and they conflict.

### PointSize*[extensions]*

An integer giving the EM square size in decipoints. For example 140 is 14-points.

For scalable typefaces, you can expand the horizontal size (set size) to make a font wider or narrower than normal for that font. The syntax for this optional extension is:

[ +*setsize* ]

+*setsize* The horizontal size in decipoints. If this field is not specified, it is assumed to be the same as PointSize.

For example, "140+240" requests a font 14 points high, and 24 points wide (or, more accurately, a 14-point font whose width is that of a 24-point font).

If neither PixelSize or PointSize are specified, the assumption is 12-point. If both are specified and they conflict, an error is returned. Use *either* PixelSize or PointSize, but not both.

ABCDEFGHIJKLMNOPQRSTUVWXYabcdefghijklmnopqrstuvwxy
ABCDEFGHIJKLMNOPQRSTUVWXYabcdefghijklmnopqrstuvwxy
ABCDEFGHIJKLMNOPQRSTUVWXYabcdefghijklmnopqrstuvwxy
ABCDEFGHIJKLMNOPQRSTUVWXYabcdefghijklmnopqrstuvwxy
ABCDEFGHIJKLMNOPQRSTUVWXYabcdefghijklmnopqrstuvwxy
ABCDEFGHIJKLMNOPQRSTUVWXYabcdefghijklmnopqrstuvwxy
ABCDEFGHIJKLMNOPQRSTUVWXYabcdefghijklmnopqrstuvwxy

**Figure 6-8. The Same Font in Different Sizes**

### ResolutionX, ResolutionY

The horizontal (X) and vertical (Y) resolution of the device that the font was designed for, measured in pixels-per-inch. If the resolution is not specified in a request XLFD name, the X server defaults to the resolution of the display for which the font is requested.

## Spacing

A code indicating the spacing between units in the font.

| | |
|---|---|
| `M` | Monospaced (fixed pitch) |
| `P` | Proportional spaced (variable pitch) |
| `C` | Character cell. The glyphs of the font can be thought of as "boxes" of the same width and height that are stacked side by side or top to bottom. |

## AverageWidth

An integer string giving the average, unweighted width of all the glyphs in the font, measured in 1/10th device-dependent pixels.

## CharSetRegistry

The registration authority that registered the specified CharSetEncoding. The XLFD conventions expect organizations that control characters to register with the X Consortium and be given a unique name to use in this field.

## CharSetEncoding*[extensions]*

The character set from which the characters in the font are drawn.

6

For scalable typefaces, this field can be used to specify subsets of any of the character sets. This is a performance hint that the X or font server uses to determine which characters need to be realized. The syntax for this optional extension is:

=*value,value...*

| | |
|---|---|
| *value* | A character or range of characters to be included in the font, specified as decimal or hex number[s]. A range is two numbers separated by a colon (:). For example, |

```
=65,0x45,90:95
```

specifies the characters "A," "E," and "P" through "U."

If an application requests a character not in the subset, then:

- A space will be substituted for that character *if* space is in the subset.
- A null, zero-width character is substituted if space is *not* in the subset.

## Using the XLFD Font Name

You use the XLFD name or alias whenever you need to specify a font. Some locations are:

- Application default or resource files, for example:

  `hpterm*Font:` *fontname*

- Command line to start clients or applications, for example:

  `xclock -digital -fn` *fontname*

- The `fonts.dir` file. Refer to "The fonts.dir File".
- The `fonts.alias` file. Refer to "The fonts.alias File".

## The fonts.dir File

In directories containing scalable and/or bitmapped fonts, the X and font servers associate each font file name with the XLFD font name by means of the `fonts.dir` file. This file is created by the installation process, or by executing the `mkfontdir` (for directories that contain only bitmapped fonts), or `stmkdirs` (for directories that contain bitmapped and/or scalable fonts) utility. The `stmkdirs` utility is described in more detail in the section on font administration.

6

You should run `mkfontdir` or `stmkdirs` after you add or delete fonts from a directory, so that the change is reflected in `fonts.dir`. You can view the list of fonts in a font directory, and their XLFD names, by examining the `fonts.dir` file in that directory.

Scalable typefaces listed in `fonts.dir` have some values set to zero.

A `fonts.dir` file looks similar to this:

```
7
helv008.pcf.Z -adobe-helvetica-medium-o-normal--8-80-75-75-P-47-hp-roman8
helvB008.pcf.Z -adobe-helvetica-bold-o-normal--8-80-75-75-P-48-hp-roman8
helvR08.pcf.Z -adobe-helvetica-medium-r-normal--8-80-75-75-P-46-hp-roman8
ant_oliv.ifo -agfa-antique olive bold-bold-r-normal-91118-0-0-0-0-p-0-hp-td00000000
ant_oliv.ifo -agfa-antique olive compact-normal-r-compact-91120-0-0-0-0-p-0-hp-td00000000
ant_oliv.ifo -agfa-antique olive italic-normal-i-normal-91846-0-0-0-0-p-0-hp-td00000000
ant_oliv.ifo -agfa-antique olive-normal-r-normal-91119-0-0-0-0-p-0-hp-td00000000
```

In this example:

- The first line lists how many bitmapped fonts and scalable typefaces are described by the file, in this case 7.
- The rest of the lines give the file name and XLFD name that describes the file.
  - The 3 lines starting with `helv ...` are 3 different bitmapped fonts. They are different versions of the "Helvetica" style made by Adobe. They are all 8-points in size, but differ in the slant and boldness.
  - The last 4 lines are 4 scalable typefaces. Several fields in the reference XLFD name are set to zero. In the request XLFD name you use to request one of these fonts, you supply values for either the PointSize or the PixelSize.

The X server tries to match your request with the bitmapped fonts and scalable typefaces listed in the `fonts.dir` file as follows: The server looks in the directories in your font path in the order shown by `xset q`.

- Bitmap fonts.

  The server uses the first font that it finds that meets all the criteria you specified in the XLFD name. If you specified everything, it will try to find the exact match. If you used the wild cards (* or ?), it will use the first font that matches the parts you did specify.

- Scalable typefaces.

  The PixelSize, PointSize, ResolutionX, and ResolutionY fields in the reference XLFD name are zero. Your request XLFD name must have either the PixelSize or PointSize (but not both). The server returns a font made from the outline with the specifications you requested.

If none of the above results in a font being returned, the X server returns an error message.

## The fonts.alias File

A font can be referred to by an alias. The alias is shorter and easier to remember (and type) than the complete font description. Aliases are found in the `fonts.alias` file. The `fonts.alias` file need not be in each directory, but the directory containing it must be in the font path.

A simple `fonts.alias` file is created as part of installing the font. The `fonts.alias` file is in this format:

> "FILE_NAME_ALIASES"
> *alias-name xlfd-name*

where:

| | |
|---|---|
| *alias-name* | is the nickname for the font. |
| *xlfd-name* | is the XLFD name that specifies the font. If the family name contains spaces, enclose the whole XLFD name in quotation marks("). |

The `fonts.alias` file provides for two types of alias names:

- The font's file name.

  If the string "FILE_NAMES_ALIASES" occurs in the `fonts.alias` file, then a font can be referred to by its file name alone, without the path name or extensions. The X server will look in all the directories in your font path.

- A name you select.

  You can specify what alias to use for referring to a font.

Any fonts not in the `fonts.alias` file must be referred to by the XLFD name.

When you edit a `fonts.alias` file, any X or font servers using that directory must be informed that they need to reread the file. To force an X server to read its `font.alias` files, run

```
xset fp rehash
```

To force a font server to read its `fonts.alias` files, run

```
kill -USR1 <pid>
```

where <pid> is the process ID of the font server.

## Using Alias Names

For example, with this `fonts.alias` file and the `fonts/hp_roman8/75dpi` subdirectory of the system directory in the font search path:

```
"FILE_NAMES_ALIASES"
ellen *-adobe-courier-bold-r-normal-*-8-80-75-75-m-50-hp-roman8
```

then you can use any of the following commands to start a digital clock using this font:

- The "FILE_NAMES_ALIASES" entry lets you use just the file name, without the path or extension.

  ```
  xclock -digital -fn CourB08
  ```

- The alias name you specified.

  ```
  xclock -digital -fn ellen
  ```

- You can always specify the XLFD name, whether or not you have a `fonts.alias` file.

  ```
  xclock -digital -fn *-adobe-courier-bold-r-normal-*-8-80-75-75-m-50-hp-roman8
  ```

- You can specify enough of the XLFD fields to identify the font characteristics you want, and represent the rest with wildcards, with 14 dashes separating the fields. The X server selects the first font in its search path that matches the specification.

  ```
  xclock -digital -fn *-*-courier-bold-r-normal-*-8-*-*-*-*-*-hp-roman8
  ```

This is useful for vendor independence—you can have the same programs and default files on different vendors' computers, and customize by making the appropriate entry in the `fonts.alias` file.

## Errors

If you get a default font or an error message (such as "can't make font ... ") when you request a font:

- Check the XLFD name for spelling.
- Check the XLFD name for inconsistencies. For instance, you cannot specify both the PixelSize and PointSize for scalable typefaces. If you think there might be a conflict, set one of the parameters to an asterisk (*) and try again.
- Run `xlsfonts` to see if the font you requested is available to you.
- Run `xset q` to see if the directory containing the font you requested is in your font search path.
- Run `xset fp rehash` to be sure the X server is using the latest aliases and font paths.

## Bitmapped Font Administration

Bitmapped fonts are included with the X Window System. They are located in the `fonts` subdirectories of the system directory. You may use them as described in the following chapters without special installation or licensing steps.

### Adding and Deleting Bitmapped Fonts

To add a bitmapped font:

1. Put the font into the `.pcf` format using `bdftopcf`.
2. Compress the `.pcf` file using `compress`.
3. Copy the file into the desired directory.
4. Run `mkfontdir` to update the `fonts.dir` file for that directory.
5. If the directory is providing fonts to the X server, run `xset -fp rehash` to notify the X server of the changes. If the directory is providing fonts to a font server, run `kill -USR1 <pid>`, to notify the font server of the changes.

To delete a bitmapped font:

1. Delete the font file.

2. Run `mkfontdir` to update the `fonts.dir` file for that directory.
3. If the directory is providing fonts to the X server, run `xset -fp rehash` to notify the X server of the changes. If the directory is providing fonts to a font server, run `kill -USR1 <pid>`, to notify the font server of the changes.

## Creating a fonts.dir file with 'mkfontdir'

The `mkfontdir` utility creates the `fonts.dir` file within a font directory.

The syntax for `mkfontdir` is:

`mkfontdir` *directory,directory,...*

where:

| | |
|---|---|
| *directory* | is a font directory. If no directory is given, the current directory is assumed. |

## Compiling BDF Fonts to PCF Fonts with 'bdftopcf'

X bitmapped fonts can be represented in several formats. A font's format is signified by the extension that appears after the font's file name:

| | |
|---|---|
| .pcf | Portable binary font description file. |
| .pcf.Z | Compressed .pcf file. |
| .bdf | Plain text font description file. |
| .bdf.Z | Compressed .bdf file. |
| .bcf | Compressed .bdf file. |
| .snf | (Prior to X11R5) Non-portable binary font description file. |
| .snf.Z | (Prior to X11R5) Compressed .snf file. |
| .scf | (Prior to X11R5) Compressed .snf file. |

Although all of these formats can be read by the X and font servers, the preferred representation for font storage are the .pcf and .pcf.Z formats. All bitmapped fonts shipped with HP-UX are shipped in the .pcf or .pcf.Z format.

The font compiler `bdftopcf` converts a font in bitmap distribution format (.bdf) into the .pcf format.

The syntax for `bdftosnf` is:

```
          ┌ -pnumber   ┐
          │ -unumber   │
          │ ⎧ -l ⎫     │
          │ ⎩ -L ⎭     │
bdftopcf  │ ⎧ -m ⎫     │ filename
          │ ⎩ -M ⎭     │
          │ -t         │
          │ -i         │
          └ -o pcf_file┘
```

where:

| | |
|---|---|
| `-p` | Specifies that font characters should be padded on the right with zeros to the boundary of word *number* where *number* is 1, 2, 4, or 8. |
| `-u` | Force the scanline unit padding to 1, 2, 4, or 8. |
| `-l` | Specifies the output of `bdftopcf` to be least significant byte first. |
| `-L` | Specifies the output of `bdftopcf` to be least significant bit first. |
| `-m` | Specifies the output of `bdftopcf` to be most significant byte first. |
| `-M` | Specifies the output of `bdftopcf` to be most significant bit first. |
| `-t` | Expand glyphs in "terminal emulator" fonts to fill the bounding box. |
| `-i` | Don't correct ink metrics for "terminal emulator" fonts. |
| `-o` | Specifies the name of the output .pcf file. If not specified, `bdftopcf` sends its output to stdout. |
| *filename* | Specifies the name of the .bdf font to be converted into portable compiled format. |

The following example takes a bitmap distribution bitmap font file named `tmrm12b.bdf` and converts it to a compressed portable compiled format file:

```
bdftopcf tmrm12b.bdf | compress >timrm12b.pcf.Z
```

## Scalable Typeface Administration

A font administrator is anyone who has purchased a font and wants to use it on a system. Scalable typefaces, unlike the X bitmap fonts, are licensed. Read your license carefully. The font administrator is responsible for ensuring that the font is used in a legal manner. The permissions for files and directories that relate to scalable typefaces have been carefully chosen to allow you to fulfill your responsibilities.

The font administrator has three main tasks:

- Install and delete scalable typefaces.
- License and unlicense devices to use typefaces.
- Add and delete character sets.

Each system is shipped with a core set of scalable typefaces installed in two different directories. Intellifont scalable typefaces from Agfa Compugraphic are installed in the directory `/usr/lib/X11/fonts/ifo.st`, and Type 1 scalable typefaces from Adobe are installed in the directory `/usr/lib/X11/fonts/type1.st`. You can list them by typing . . .

```
xlsfonts -fn "*-0-0-0-*"
```

. . . as described earlier in this chapter.

### Overview

There are four steps a font administrator must perform to make a scalable typeface ready for use. These steps are covered in more detail in the following sections.

1. Load the typeface into a directory on the target system ("Installing and Licensing Scalable Typefaces").
2. Load the character set if it is not already on the system ("Adding and Removing Character Sets"). Character sets are used by several different typefaces.
3. Run `stmkdirs` for that directory to notify the X server of the addition. ("Creating *.dir Files with 'stmkdirs'")
4. Add the license to that typeface for the system ("Adding and Removing Licenses with 'stlicense'").

An example of installation and removal of a typeface and its license is presented later in this chapter.

When you install a new scalable typeface, any running font servers that are using that directory need to be told to read the directory for the new font or fonts. You can do this by entering `kill -USR1 <pid>`, where <pid> is the process ID of the font server. See the section on using the font server for more detailed information on font server configuration.

## Installing and Licensing Scalable Typefaces

To install a scalable typeface onto a system:

1. Decide what directory will contain the new typefaces.
   - If you use the `fonts/ifo.st/` or `fonts.type1.st` subdirectories of the system directory, you will have your fonts in centralized locations, but you need superuser capability to write in either of these directories.
   - If you create your own directory, you do not need superuser capability. If you create a new directory, be sure to:
     - give it the extension `.st`.
     - make it readable for the group `bin`.
     - configure the font server to include it in its font path.

2. Create a "typefaces" subdirectory to the `.st` directory, if one does not already exist. Install new fonts into this directory.

   **For Intellifont format fonts:**

   a. Copy the files containing the typeface into an empty temporary directory on the target file system.

   b. If your typefaces are contained on several flexible discs, load the entire contents of each disc into its own temporary directory *or* do these steps for each individual disc. Copy the entire contents of the disc, even if you want only one typeface from it.

**Caution** If you copy all the discs into one directory, some files will be overwritten.

- For HP-UX media, copy the files directly to the temporary directory.

- For MS-DOS media, use the `doscp` utility to copy the files from a flexible disc drive to the temporary directory.
- If a PC is networked into your system, refer to the network documentation about how to copy files from the PC to the HP-UX system.

c. Run the `stload` utility on each temporary directory to convert the files into the proper format and place the typeface in the permanent directory you established in step 1.

d. If you loaded more typefaces than you wanted, remove the file(s) and run `stmkdirs` for that directory.

e. Delete the temporary directories used in step 2b.

**For Type-1 format fonts:**

**Note** The font server can handle IBM-format Type 1 fonts, but it cannot read the Macintosh format.

a. Copy the desired scalable font files (extensions `.pfa` and `.pfb`) from the distribution media into the `typefaces` subdirectory. The suffixes *must* be lower case; the system does not recognize `.PFA` or `.PFB`.

6

b. Run `stmkdirs` for the `typefaces` subdirectory to add the new fonts to the database.

Once the file is loaded, typefaces can be made available to users through licensing. Refer to “Adding and Removing Licenses with ‘stlicense’”.

To delete a typeface from a system:

1. Remove all licenses for the product, using `stlicense`. For example

```
stlicense -pr foo -fp /users/ellen/ifo.st "*"
```

   removes all licenses to product “foo” in the specified directory.

2. Remove the typeface files that are no longer being used (extensions `.ifo`, `.pfa`, and `.pfb`) from the `typefaces` subdirectory.
3. Run `stmkdirs` in the typefaces subdirectory to update the `fonts.dir` file.
4. Remove the product file from the `products` subdirectory.

## Loading Scalable Typefaces with 'stload'

Use the `stload` utility to load into the system Agfa Compugraphic fonts that have been distributed in Agfa's FAIS distribution format.

The syntax for the `stload` utility is:

`stload` [*options*] [*directory\filespec*]

where

| | |
|---|---|
| *directory\filespec* | Required parameter specifying the name of the directory or filespec of the data to be loaded. |
| `-o` *path* | The name of the directory to which the the typeface outlines should be written. If this is omitted, the default is to the `/fonts/if2.td/typefaces` subdirectory of the system directory. |
| `-fp` *path* | The name of the base directory under which `typefaces`, `metrics`, and `products` directories should be used. |
| `-p` *product-number* | Associates a product number with the newly-loaded typeface. Although this could be anything, it should reflect the product number on the package and media. The `stlicense` utility requires this product number. |
| `-list` | Prints a list of the data located in *directory*. |
| `-link` | Make links to the original *directory*, rather than copies. |
| `-sym` | Make symbolic links to the original *directory*. |
| `-`*id*`[,-`*id* ... ] | Identifies one or more specific typefaces to be loaded. |
| `-tfm` | Updates `.tfm` files in the output directory. |
| `-dos` | Specifies that the typeface file is in DOS format. `stload` converts DOS files into HP-UX files. |
| `-d` *mapdir* | Specifies the directory containing the symbol list map required by the `-to` option. |
| `-to` *format* | Specifies the symbol list that should be used for assigning character ID codes when loading FAIS data. |
| `-f` *libname* | Specifies the name of the library into which FAIS data should be loaded. |

| | |
|---|---|
| `-u` | Specifies that the `.dir` files *not* be updated. |
| `-v` | Specifies verbose mode. |
| `-h` | Requests help. |

For example:

```
   stload -fp new.st -p C2054#ABA -tfm -dos -v tempdir
```

## Creating *.dir Files with 'stmkdirs'

Use the `stmkdirs` utility to create and maintain various configuration files that support the scalable typeface technology, including `fonts.dir` files, directories of character sets, and directories of metrics files.

This section describes how to use `stmkdirs` to maintain font directories. `stmkdirs` works much like `mkfontdir`, in that it creates the `fonts.dir` file containing a list of fonts in the current directory. But, `stmkdirs` recognizes Intellifont and Type 1 scalable fonts, while `mkfontdir` recognizes only bitmapped fonts. Therefore, you should use `stmkdirs` to build the `fonts.dir` file in any font directory that contains scalable fonts.

`stmkdirs` creates the `fonts.dir` file from the directory of font files.

`stmkdirs` [*options*] *directory* [ ,*directory,...* ]

where the options are:

| | |
|---|---|
| `-tfm` | For any Intellifont files in the target directories, build a TFM (Tagged Font Metrics) file in the specified destination directory. |
| `±m` | Requests that `fonts.dir` be generated including (+) or excluding (-) bitmap fonts. |
| `±o` | `fonts.dir` is generated including (+) or excluding (-) `.ifo` libraries. |
| `±f` | `fonts.dir` is generated (+) including excluding (-) both `.ifo` libraries and bitmap libraries, or not generated (-). |
| `±c` | Requests that `charsets.dir` be generated (+) or not generated (-). |
| `-b` | Suppresses creation of backup files. |
| `-h` | Prints help information on stout. |
| *directory* | is one or more directory names containing fonts. |

For each directory listed, `stmkdirs` reads all the font files in that directory, putting file names and XLFD name into the `fonts.dir` file. Without a `fonts.dir` file, the the X and font servers cannot access font files in the directory.

Run `stmkdirs` after any fonts or charsets are added or deleted.

## Adding and Removing Licenses with 'stlicense'

When you purchase a scalable typeface product, you receive a license agreement, outlining by who and how the typefaces in the product may be used. For instance, the terms may be that only one printer and one display may use the typefaces.

The `stlicense` utility helps administer the licenses. Fonts are available only to licensed devices.

The syntax for the `stlicense` utility is:

$$\texttt{stlicense}\ [\texttt{-fp}\ \textit{directory}] \left\{ \begin{array}{l} \texttt{-fn}\ \textit{typeface} \\ \texttt{-pr}\ \textit{product} \end{array} \right\} [\pm\textit{device} \ldots\ ]$$

where

| | |
|---|---|
| `-fp` | The path of directories to search for the specified product or typeface. The default is `/fonts/ifo.st/` in your system directory. |
| `-fn` | The *typeface* being licensed. The typeface is specified as an XLFD name. You need not use the whole XLFD name, just enough to uniquely identify the typeface. A product is identified by its name or product number. |
| `-pr` | The *product* being licensed. |
| ±*device* | The *device* is specified in the form: `host:device`. The given typeface is added to or removed from the list of typefaces licensed for this device.<br><br>The host name `STSYSTEM` refers to all hosts served by this typeface directory.<br><br>The device name `DISPLAYS` refers to all displays running on the host.<br><br>The device name `PRINTERS` refers to all printers connected to the host. |

| | |
|---|---|
| | If the machine is not specified, the default is the machine on which `stlicense` is running, and the device defaults to `DISPLAYS`. |
| nothing | If no devices are given, a list of devices that have licenses for the typeface is printed on the standard output. The list is grouped by system and individual device licenses. |

The built-in typefaces are licensed at installation time to all displays and printers attached to the system (`STSYSTEM:DISPLAYS` and `STSYSTEM:PRINTERS`). For example,

```
stlicense -pr C2054#ABA +lj3
```

licenses the printer named lj3 to use the typeface product C2054#ABA. Since the machine is not specified, `stlicense` assumes the machine to be the one on which it is running.

```
stlicense C2054#ABA -pr -laserjp +laserkb
```

## Adding and Removing Character Sets

6

Many Intellifont and Type 1 scalable fonts contain many characters, and can be used to create more than one character set. For example, both Intellifont and Type 1 scalable fonts can be used to build fonts using either ISO8859 encoding or HP roman-8 encoding. This section describes the management of character sets for scalable fonts.

**Administering Character Sets for Intellifont Fonts.** Character set definitions are stored in the `fonts/stadmin/ifo/charsets` subdirectory of the system directory as ASCII files with the extension `.sym`. The charsets directory is shipped from the factory with two popular character sets definitions:

- HP Roman 8
- ISO 8859-1 (also known as ECMA Latin 1)

These character sets are the only ones many applications need.

The `archive` subdirectory contains definitions for a number of additional character sets. These include character sets popular for PCs.

To enable one of the character sets in "archive":

1. Copy the desired character set (`.sym`) file from the `archive` subdirectory into the `charsets` directory. For example,

   ```
   cp /usr/lib/X11/fonts/stadmin/charsets/archive/pc8.sym ..
   ```

2. Run `stmkdirs` in the `charsets` directory to update the `charsets.dir` file. For example,

   ```
   stmkdirs /usr/lib/X11/fonts/stadmin/charsets
   ```
3. Run the following to notify the font server of the changes:

   ```
   kill -USR1 <pid>
   ```

   where <pid> is the process ID of the font server.

To install a character set from the Type Director/DOS product, first run `stconv` on the `.sym` file to put it into a format that can be used on your workstation.

To delete a character set:

1. Remove the character set (`.sym`) file from the `charsets` directory. (It is still in the `archive` subdirectory if you need it later.)
2. Run `stmkdirs` with the `+c` option in the `charsets` directory to update the `charsets.dir` file.
3. Run the following to notify the font server of the changes:

   ```
   kill -USR1 <pid>
   ```

   where <pid> is the process ID of the font server.

**Administering Character Sets for Type 1 Fonts.** Character set definitions for Type 1 fonts are stored in `/usr/lib/X11/fonts/stadmin/type1/charsets`. Files in this directory named `cp.`<character_set> define the character mapping for the desired character set. The two files shipped in this directory, `cp.iso8859` and `cp.hp-roman8`, define the character set mappings for ISO8859.1 and Roman-8 encoding.

To add or delete character set mappings for Type 1 fonts, you need to add or delete mapping files to this directory. The file names must be of the form `cp.`<character_set>, where <character_set> is the charset definition, containing one hyphen, to be used at the end of the font's XLFD name.

When a font server starts up or rereads its font directories in response to a signal, it uses the character sets defined in this directory to build its list of available font names.

## Example: Installing and Licensing

This example shows installing and licensing an Intellifont typeface product called "COOOO#AAA". Path names are shown in full for clarity, you may not need to specify them in that detail.

"COOOO#AAA" is the product number on the box of the product. It comes on two flexible discs.

A new scalable typeface directory is to be created. It is owned by the font administrator, /users/ellen. A flexible disc drive is attached to the system at device location /dev/rdsk/2s1.

1. Copy each of the two discs into its own temporary directory.

   ```
   mkdir /tmp/disc1
   ```
   *insert flexible disc 1 into the drive.*
   ```
   doscp /dev/rdsk/2s1/* /tmp/disc1
   mkdir /tmp/disc2
   ```
   *insert flexible disc 2 into the drive.*
   ```
   doscp /dev/rdsk/2s1/* /tmp/disc2
   ```

2. Create a new directory for the scalable typeface and make it readable by the bin group. All other groups should have no access to the .ifo files.

   ```
   mkdir /users/ellen/new.st
   chacl "%.bin+r" /users/ellen/new.st
   mkdir /users/ellen/new.st/typefaces
   chacl "%.bin+r" /users/ellen/new.st/typefaces
   ```

3. Load the typefaces into the new directory. Note that this example includes the creation of .tfm files. If you have applications that utilize AutoFont Support, you will need them. Otherwise, save installation time and disc space by not requesting them.

   ```
   stload -p C0000#AAA -dos -v -fp /users/ellen/new.st -tfm /tmp/disc1
   stload -p C0000#AAA -dos -v -fp /users/ellen/new.st -tfm /tmp/disc2
   ```

4. Make the new files readable by the bin group.

   ```
   chacl "%.bin+r" /users/ellen/new.st/typefaces/*"
   ```

5. Clean up the temporary directories.

   ```
   rmdir /tmp/disc1
   rmdir /tmp/disc2
   ```

6. Have your users add the new directory to their font paths.

   ```
   xset fp+ /users/ellen/new.st
   ```

7. Before this product can be used, it must be licensed. For this example, the license in the product stipulates that the typefaces can be used for up to three printers and any number of displays connected to the system.

```
stlicense -fp /users/ellen/new.st -pr C0000#AAA +STSYSTEM:DISPLAYS \
   +mysystem:laser1 +mysystem:laser2 +mysystem:laser3
```

Notice that although the printers are listed individually, the displays are grouped by the shortcut STSYSTEM:DISPLAYS. mysystem is one of the hosts covered by STSYSTEM.

If you now wanted mysystem:laser4 to be licensed, you have to remove the license for one of the other printers, since you are only allowed up to three printers.

```
stlicense -fp /users/ellen/new.st -pr C0000#AAA -mysystem:laser3 \
                                                 +mysystem:laser4
```

When the product is no longer needed, remove it from the system.

1. Remove all licenses to the product.

   ```
   stlicense -fp /users/ellen/new.st -pr C0000#AAA "-*"
   ```

2. Remove the typeface files (`.ifo`). The list of files to be removed is in `/users/ellen/new.st/products/C0000#AAA`.

   ```
   rm /users/ellen/new.st/typefaces/12345678.ifo
   rm /users/ellen/new.st/typefaces/22345678.ifo
   ⋮
   ```

3. Update the `fonts.dir` in the `typefaces` subdirectory.

   ```
   stmkdirs /users/ellen/new.st/typefaces
   ```

## Scalable Typefaces File Structure

This section describes the default scalable font directories (font catalogs). There can be other font catalogs, but each must have the `.st` extension and structure described here. In addition, each must be on the font path.

The directories described here are subdirectories of the system directory.

### Scalable Font Directories

The `fonts/ifo.st` and `fonts/type1.st` are the default font catalogs. They contain typeface files, licensing, and metrics information.

**Licenses Subdirectory.** The `licenses` subdirectory contains files with licensing information for each host, display, and system.

It contains a hosts.dir file, which is a cross-reference between the actual host name and the directory containing license information about that host. One host subdirectory is STSYSTEM, which is for system-wide licenses. There are separate subdirectories for each host on the system.

Within each host subdirectory, there are subdirectories for each device (DISPLAYS is always one). Within these directories there are fonts.dir and fonts.alias files as described elsewhere in this manual.

**Metrics Subdirectory.** The fonts/ifo.st/metrics directory contains metrics for the fonts and scalable typefaces that are not loaded on the system. This is the recommended location for the .tfm files for Intellifont fonts, and for .afm files for Type 1 fonts.

**Products Subdirectory.** Each product that has been installed has a file cross-referencing the font file name and the XLFD name used to refer to it. The core fonts are in the builtin file.

**Typefaces Subdirectory.** The typefaces subdirectory contains the typeface files. The Intellifont files have a .ifo extension. Type 1 typeface files have a .pfa or .pfb extension. In addition, there is a fonts.dir file for each typeface directory.

## Administrative Directories

The fonts/stadmin/ifo directory contains typefaces.dir, which provides a cross-reference between the typeface ID and the XLFD name for Intellifont fonts.

The fonts/stadmin/ifo/charsets subdirectory contains valid character sets for Intellifont fonts. These files have a .sym extension and are in the same format at those for TypeDirector/DOS 3.0. A charsets.dir file provides a cross-reference between the file name and the character set name. Non-active character sets are contained in the subdirectory archive, with its own charsets.dir file. To make an inactive character set active, copy it from the archive subdirectory, and update the charsets.dir file by running stmkdirs on that directory.

The fonts/stadmin/type1/charsets subdirectory contains valid character sets for Type 1 fonts. These files are all named cp.<character_set>, and are used to provide a mapping between internal Type 1 character names and standard encodings such as ISO8859.1 and Roman-8. The directories do not have to be physically present on every system. There may be a master copy on one system and NFS links to other systems. Font administration should be performed on the NFS server for the typeface directories.

## Using ‘stmkfont’ and ‘stconv’

Two additional support utilities are provided for use with Intellifont outline fonts. The next two sections describe `stmkfont`, a utility for generating a variety of bitmap formats from Intellifont outlines, and `stconv`, a utility for manipulating Intellifont symbol set files.

### Making Bitmapped Fonts from Scalable Typefaces with ‘stmkfont’

If you use a particular scalable typeface often, it is more efficient to turn it into a bitmapped font, rather that recreating it each time you want it.

The `stmkfont` utility produces bitmapped fonts in a variety of formats from an outline specified by an XLFD name. `stmkfont` can create bitmap fonts in the following formats:

| | |
|---|---|
| bdf | Bitmap Distribution Format |
| PCL | Printer Command Language (for HP LaserJet printers) |
| PCLEO | Printer Command Language Encapsulated Outlines (for HP LaserJet III printers). |
| IFO | Intellifont outline. |
| TFM | HP Tagged Font Metric for metrics pertaining to HP LaserJet printer scalable typefaces. |

The syntax for `stmkfont` is:

`stmkfont` [*options*]*xlfdname*

where the options are:

| | |
|---|---|
| `-d1`*path* | Specifies the primary database tree path (default is `fonts/ifo.td`). |
| `-d2` *path* | Specifies the secondary database tree path (default is `fonts/td`). |
| `-dv` *device* | Specifies the device for which the font is to be made. |
| `-cp` *path* | Specifies the charset path (default is `charsets`). |
| `-cf` *file* | Specifies the charset file (default is to derive it from the XLFD name. |
| `-nf` *file* | Specify a new name for `fonts.dir`. |
| `-ns` *file* | Specify a new name for `charsets.dir`. |
| `-nt` *file* | Specify a new name for `typefaces.dir`. |

|  |  |
|---|---|
| -nv *name* | Specify an environment variable to use instead of STPATH. |
| -o *outfile* | Specifies output file (default is stdout) |
| -f *format* | Specifies the output format (BDF (default), PCL, or PCLEO). |
| -I | Send completion status information to stderr. |
| -P | Send 1% progress dots to stderr. |
| -C | Send catalog of XLFD/symbol set combinations to stderr. |
| -T | Bypass intermediate tempfile, write to output directly. |
| -V | Send fully qualified XLFD name to stderr, then quit. |
| -v | Send fully qualified XLFD name to stderr, then continue. |
| -w | Suppress bitmaps, restrict output to header and trailer only. |
| -q | Suppress error messages (quiet mode). |
| *xlfdname* | This parameter is required. Since both the XLFD name and parameters start with a dash (-), then `stmkfont` assumes the last arguments is the XLFD name. |

The XLFD name must not contain any blanks. If it does, enclose the entire string in quotes ("). Empty fields and wildcards are permitted.

6

For example,

```
stmkfont -o myfont -x "-agfa-cg century schoolbook-normal-r-normal---240---p-150-*-roman8"
```

## Converting Map Formats with 'stconv'

The `stconv` utility converts symbol set maps (`.sym` files) from one symbol set to another. Output is always to `stdout`.

The syntax for `stconv` is:

`stconv` *infile* [`-hmq`] [`-d` *mapdir*] [`-to` *format*]

where:

|  |  |
|---|---|
| *infile* | Name of the `.sym` file to be converted. |
| -d *mapdir* | Specifies the name of the directory containing the symbol conversion list. This directory should contain the file `acg.hpmsl` and any optional additional symbol set maps. The default is `/fonts/stadmin/ifo/charsets` in your system directory. |

| | |
|---|---|
| | default is `/fonts/stadmin/ifo/charsets` in your system directory. |
| `-to` *format* | Specifies the new symbol list format. The default is `hpmsl`. To generate a symbol set for Agfa Compugraphic's character codes, specify `-to ACG`. |
| `-m` | Lists the conversion map. |
| `-q` | Run quietly. |
| `-h` | Requests help. |

For example,

```
stconv -to ACG roman8.sym
```

reads the HPMSL symbol map `roman8.sym`, and writes the AGC version to stdout.

## Using Native Language Input/Output

Though most character sets are composed of 8-bit characters, some languages (Japanese, Chinese, and Korean) have larger character sets that require 16-bit characters. The X Window System supports the use of 16-bit character input with the Native Language Input/Output (NL I/O) subsystem.

### Requirements for Using NL I/O

To use NL I/O you must have the following:

- The NL I/O subsystem properly installed on your system.
- The appropriate language keyboard *or* an ASCII keyboard. A client that uses `XHPSetKeyboardMapping` allows NL I/O to be used with any language HP keyboard.

  X clients use the `KBD_LANG` environment variable to determine which keyboard language to pass to `XHPSetKeyboardMapping`.
- The appropriate NL I/O fonts installed in the `fonts` subdirectory of your system directory:

  ```
  fonts/hp_japanese/
  fonts/hp_chinese_s/
  ```

```
    fonts/hp_chinese_t/
    fonts/hp_korean/
    fonts/hp_katakana/
```

For more information on native language configuration, refer to “Customizing for Native Language Support” in chapter 3.

## Specifying an NL I/O Font

NL I/O fonts are part of the NL I/O product. They are installed in the subdirectories of `fonts` subdirectory of your system directory when you install your NL I/O subsystem.

You specify an NL I/O font exactly like you specify any other font. For example, if you want to create an `hpterm` window that uses the Japanese font `jpn.8x18`, use the following command line:

```
hpterm -fn jpn.8x18
```

7

# The Window Manager

The OSF/Motif Window Manager (mwm) is an X11 client that manages the appearance and behavior of objects on the root window. You control mwm and its management functions using a mouse, keyboard, and a functional window frame. Additionally, mwm has a root menu to assist you in the control of the root window.

Chapter 4 explains how to *use* windows. This chapter explains how to customize them.

This chapter organizes window manager resources and functions into the following task-oriented topics:

- Starting and stopping mwm.
- Setting mwm resources using .mwmrc
- Managing the general appearance of window frames.
- Working with icons.
- Managing window manager menus.
- Using the mouse.
- Using the keyboard.
- Controlling window size and placement.
- Controlling focus policies.

## Starting and Stopping the Window Manager

The OSF/Motif Window Manager (`mwm`) is an X11 client that manages the appearance and behavior of objects on the root window. You control `mwm` and its management operations using a mouse, a keyboard, and a functional window frame. Additionally, `mwm` has a root menu to assist you in the general control of the root window.

The OSF/Motif Window Manager is the default window manager for your X Window System. It is started from `$HOME/.x11start` when you start X11. If that file doesn't exist, `mwm` is started from `sys.x11start` in your system directory.

The syntax for `mwm` is as follows:

```
     ┌-display host:display.screen┐
     │-xrm resourcestring         │
mwm  │-multiscreen                │
     │-name name                  │
     └-screens name [name ...]    ┘
```

where:

| | |
|---|---|
| `-display` | Specifies the screen to use. |
| `-xrm` | Specifies using the named resource on starting. |
| `-multiscreen` | Causes `mwm` to manage all screens on a display. The default is to manage only a single screen. |
| `-name` | Uses *name* to retrieve resources. |
| `-screens` | Gives the resource names for the screens managed by `mwm`. The names are separated by spaces. |

The following line in `.x11start` in your home directory starts `mwm`.

```
mwm $@ &
```

The `$@` passes the window manager options specified on the `x11start` command line.

## Declaring Resources

The mwm client receives configuration information from three resource files:

- sys.Xdefaults in the system directory or .Xdefaults in your home directory.

  Contains X resources.

- system.mwmrc in the system directory or .mwmrc in your home directory.

  Menus, key bindings, and button bindings.

- app-defaults/Mwm in the system directory.

  X resources for mwm only.

  This file cannot be changed. However, you can copy information from that file, modify it, and then add it to your personal resource.

If you modify these files, you can use either method of specifying personal resources: changing the RESOURCE_MANAGER property or modifying the .Xdefaults file. Both methods are covered in chapter 5.

The syntax you use differs depending on whether you want the resource to control an element or that element *for a particular object.*

The syntax for mwm resources is:

$$\texttt{Mwm*}\left[\left\{\begin{array}{l}\textit{clientname clientclass}\\ \texttt{defaults}\end{array}\right\}\right]*\textit{resource}\texttt{:}\ \textit{value}$$

Use nothing between "Mwm" and the resource name if you want the resource applied to all clients for which you don't otherwise specify a value. Some resources make sense only at this level, such as the focus policy ones. Use *clientclass* to apply the resource to a specific class of clients. Use *clientname* to apply the resource only to a specific instance of a client named using the client's name resource. Use defaults when you want the default value used.

For example, if you want the general appearance of the clients in your environment to be SteelBlue and VioletRed, but want your menus to be different, you could use the following lines in your personal resources.

```
Mwm*background:         SteelBlue
Mwm*foreground:         VioletRed
Mwm*activeBackground:   VioletRed
Mwm*activeForeground:   SteelBlue

Mwm*menu*background:    SkyBlue
Mwm*menu*foreground:    White
```

Or, if you want to use your own `happyface` bitmap for `hpterm` windows and see a complete label whenever any icon is active, you would have the following lines in your personal resources:

```
Mwm*HPterm*iconImage: /users/yourusername/Bitmaps/face.bits
Mwm*iconDecoration: label activelabel
```

7

## Frames

You can control the general appearance of the window frames in your environment with your personal resources specifications.

### Parts of a Window Frame

Three aspects of the general appearance of window frames are under your control.

| | |
|---|---|
| Color | The color of foreground, background; and top, bottom, and side shadows. |
| Tile | The mixture of foreground and background color that composes the pattern of the frame surface. |
| Font | The style (including size) of the text characters in the title bar, menus, and icon labels. |

Additionally, you can control what parts of the frame are displayed.

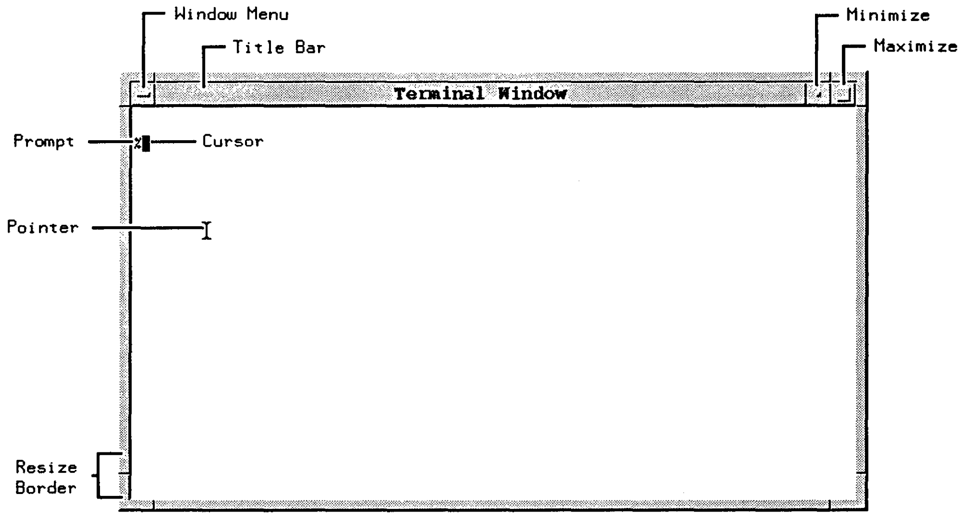


Figure 7-1. Frame Elements

## Customizing the Window Frames

You can specify what frame components you want to appear on windows:

- The `clientDecoration` resource enables you to choose just how much or how little "decoration" you want to put around each client.
- The `transientDecoration` resource enables you to choose just how much or how little decoration you want to put around each transient window. A **transient window** is a relatively short-lived window, for example, a dialog box.

You can still access the functionality of any decoration you remove by binding its functions to mouse buttons or to key presses, as explained in "Mouse Button Bindings" later in this chapter.

**Table 7-1. Valid Window Frame Elements**

| Frame Element | Description |
|---|---|
| all | Include all frame elements (default value). |
| none | Include no window frame elements. |
| ±border | Window border. |
| ±maximize | Maximize button (includes title bar). |
| ±minimize | Minimize button (includes title bar). |
| ±none | Include no window frame elements. |
| ±resizeh | Resize border handles (includes border). |
| ±menu | Window menu button (includes title bar). |
| ±title | Title bar. |

7

You specify the `clientDecoration` and `transientDecoration` resources as a list of the frame elements. If the first element in the list is preceded by a plus (+) sign or has no sign preceding it, the window manager starts with no frame and assumes that the list contains those elements you want added. If the list begins with a minus (–) sign, the window manager starts with a complete frame and assumes that the list contains elements you want removed from the frame.

For example, you may want a border with only a title bar and window menu button around a particular hpterm window started as hpterm -name hp850.

```
   Mwm*hp850*clientDecoration: +menu
```

Or you could remove the title bar from all transient windows by adding the following line in your personal resources specification:

```
   Mwm*transientDecoration: -title
```

## Coloring Window Frame Elements

You can use any of the standard X11 colors listed in the rgb.txt file in your system directory to color frame elements. In addition, you can create your own colors using hexadecimal values (see "Color Resources" in chapter 5).

The following table lists the individual elements of inactive and active window frames, and the resources that control their color, for mwm.

The default settings provide a 3-D visual effect without you having to specify the exact colors for every frame element.

**Table 7-2. Window Frames Resources for a Color Display**

| To color this ... | Use this resource ... | The default value is ... |
|---|---|---|
| Background of inactive frames. | `background` | LightGrey |
| Left and upper bevel of inactive frames. | `topShadowColor` | Lightened `background` color |
| Right and lower bevel of inactive frames. | `bottomShadowColor` | Darkened `background` color |
| Foreground (title bar text) of inactive frames. | `foreground` | Darkened `bottomShadowColor` |
| Background of the active frame. | `activeBackground` | CadetBlue |
| Left and upper bevel of the active frame. | `activeTopShadowColor` | Lightened `activeBackground` color |
| Right and lower bevel of the active frame. | `activeBottomShadowColor` | Darkened `activeBackground` color |
| Foreground (title bar text) of the active frame. | `activeForeground` | Darkened `activeBot-tomShadowColor` |

For example, the following lines in the `.Xdefaults` file in your home directory give the window manager frame a maroon foreground and a gray background. The background color is used to generate colors for the top and bottom shadow elements so that a 3-D effect is achieved.

```
Mwm*foreground: Maroon
Mwm*background: Gray
```

## Tiling Window Frames With Pixmaps

A **pixmap** can be used to create shades of colors. Each pixmap is composed of tiles. A **tile** is a rectangle that provides a surface pattern or a visual texture by "mixing" the foreground and background colors into a color pattern.

**Table 7-3.**
**Tiling Window Frames with Window Manager Resources**

| To tile this ... | Use this resource ... | The default for color displays is ... |
|---|---|---|
| Background of inactive frames. | `backgroundPixmap` | NULL |
| Right and lower bevels of inactive frames. | `bottomShadowPixmap` | NULL |
| Left and upper bevels of inactive frames. | `topShadowPixmap` | NULL |
| Background of the active frame. | `activeBackgroundPixmap` | NULL |
| Right and lower bevels of the active frame. | `activeBottomShadowPixmap` | NULL |
| Left and upper bevels of the active frame. | `activeTopShadowPixmap` | NULL |

The following table lists the acceptable values for pixmap resources:

**Table 7-4. The Values to Use for Tiling Window Frames**

| To tile an element this color ... | Use this value ... |
|---|---|
| The foreground color. | foreground |
| The background color. | background |
| A mix of 25% foreground to 75% background. | 25_foreground |
| A mix of 50% foreground to 50% background. | 50_foreground |
| A mix of 75% foreground to 25% background. | 75_foreground |
| In horizontal lines alternating between the foreground and background color. | horizontal_tile |
| In vertical lines alternating between the foreground and background color. | vertical_tile |
| In diagonal lines slanting to the right alternating between the foreground and background color. | slant_right |
| In diagonal lines slanting to the left alternating between the foreground and background color. | slant_left |

The following figure illustrates the valid tile values:

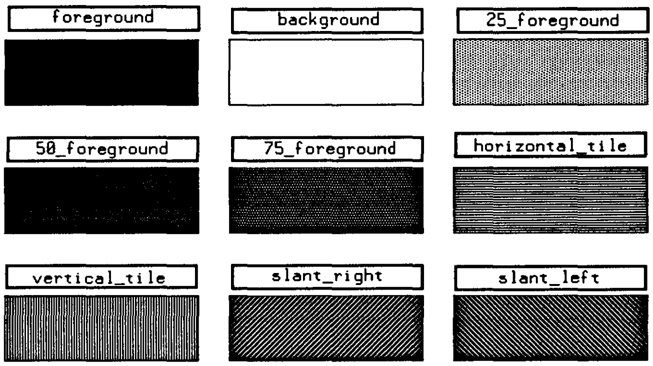


**Figure 7-2. Valid Tile Values**

## Matting Clients

A **matte** is a 3-D border just inside the window between client area and window frame.

The following table lists matte elements and the resources that control their color.

7

**Table 7-5.**
**Coloring Window Frames with Window Manager Resources**

| To color this ... | Use this resource ... | The default value is ... |
|---|---|---|
| Width of matte | matteWidth | 0 (no matte) |
| Matte background. | matteBackground | mwm background |
| Left and upper bevel of matte. | matteTopShadowColor | Lightened matteBackground color. |
| Right and lower bevel of matte. | matteBottomShadowColor | Darkened matteBackground color. |
| Matte foreground. | matteForeground | Darkened matteBot-tonShadowColor. |
| Matte right and lower bevels. | matteBottomShadowPixmap | client bottom shadow color |
| Matte left and upper bevels. | matteTopShadowPixmap | client top shadow color |

The values to use for tiling mattes are shown in Table 7-4.

For example, you could place a different matte around all instances of hpterm and xterm windows by including the following lines in your personal resources specifications:

```
Mwm*HPterm*matteWidth:          10
Mwm*HPterm*matteBackground:     SkyBlue
Mwm*XTerm*matteWidth:           10
Mwm*XTerm*matteBackground:      Tan
```

## Frame Resources For Monochrome Displays

If mwm determines that the monitor is monochrome, and no color resources are specified for frame elements, mwm uses defaults appropriate for monochrome displays. Mwm*background and Mwm*activeBackground are set to White. The following table lists the frame elements, resources, and defaults for monochrome monitors.

**Table 7-6.**
**Window Frame Resource Values for Monochrome Monitors**

| The background is ... | For this resource ... | The default value is ... |
|---|---|---|
| White | topShadowColor | White |
| White | bottomShadowColor | Black |
| White | foreground | Black |
| White | topShadowPixmap | foreground |
| White | activeBackgroundPixmap | foreground |
| White | activeTopShowdowPixmap | 50_foreground |

The sys.Xdefaults file contains a set of entries that provides a more attractive window shading for monochrome displays. These entries start with mwm_bw, and require that you start mwm with the name mwm_bw. To do this, edit the following line in .x11start:

```
mwm & #Starts the mwm window manager
```

to read:

```
mwm -name mwm_bw & #Starts the mwm window manager
```

You must restart X11 in order for this change to take effect.

When you start the window manager with a new name, it will no longer see resources of the form mwm**resource*. It will see the class resources Mwm**resource*.

## Specifying a Different Font for the Window Manager

The default font for the text of the OSF/Motif Window Manager is the `fixed` font. However, you can use the `fontList` resource to specify a different font if you desire. The `fontList` resource can use any valid X11 font name as its value. For more information about fonts, see chapter 6.

## Working with Icons

Icons provide a handy way to straighten up a cluttered workspace.

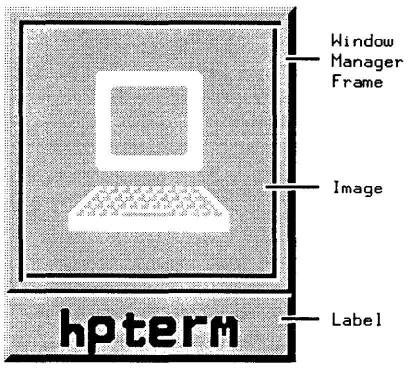


**Figure 7-3. The Parts of an Icon**

An icon image (a bitmap) is the actual graphic illustration of the icon. An image can come from any one of the following three sources, listed in order of precedence:

| | |
|---|---|
| user | You, the user, can specify an icon image using the `iconImage` resource. |
| client | A client can use the `WM_HINTS` window property to specify either an icon window or a bitmap for the window manager to use as the icon image. |
| default | The window manager will use its own built-in default icon image if an image is not specified elsewhere. |

The window manager uses the following default order of precedence in choosing an icon image:

The resource `useClientIcon` lets you interchange the precedence of user-supplied icon images and client-supplied icon images. The default value is "False." When the resource is set to "True," client-specified icon images have precedence over user-supplied icon images.

## Controlling Icon Placement

By default, the window manager places icons in the lower left corner of the root window. Successive icons are placed in a row proceeding toward the right. Icons are prevented from overlapping. An icon will be placed in the position it last occupied if no icon is already there. If that place is taken, the icon will be placed at the next free location.

The following three resources enable you to control the placement of icons:

**Table 7-7.
Controlling Icon Placement with Window Manager Resources**

| To specify this ... | Use this resource ... | The default value is ... |
|---|---|---|
| A placement scheme for icons. | `iconPlacement` | left bottom |
| The distance between screen edge and icons. | `iconPlacementMargin` | the default space between icons |
| Automatic icon placement by the window manager. | `iconAutoPlace` | True |

The following table lists the icon placement schemes available to you:

**Table 7-8. Schemes for Automatic Placement of Icons**

| If you want this icon placement ... | Choose this scheme ... |
|---|---|
| From left to right across the top of the screen. | `left top` |
| From right to left across the top of the screen. | `right top` |
| From left to right across the bottom of the screen. | `left bottom` |
| From right to left across the bottom of the screen. | `right bottom` |
| From bottom to top along the left of the screen. | `bottom left` |
| From bottom to top along the right of the screen. | `bottom right` |
| From top to bottom along the left of the screen. | `top left` |
| From top to bottom along the right of the screen. | `top right` |

For example, if you want automatic placement of icons starting at the top of the screen and proceeding down the right side, you would have the following lines in your personal resource specifications:

```
Mwm*iconPlacement: top right    Specifies the placement scheme.
Mwm*iconAutoPlace: True         Specifies automatic placement.
```

## Controlling Icon Appearance and Behavior

mwm offers you a number of resources to control the specific appearance and behavior of icons.

### Selecting Icon Decoration

Using the iconDecoration resource, you can select exactly what parts of an icon you want to display:

**Table 7-9. The Values That Control the Appearance of Icons**

| If you want an icon that looks like this ... | Use this value ... |
|---|---|
| Just the label. | `label` |
| Just the image. | `image` |
| Both label and image. | `label image` |
| The label of an active icon isn't truncated. | `label activelabel` |

### Sizing Icons

Each icon image has a maximum and minimum size. mwm has both default sizes as well as maximum and minimum allowable sizes.

**Table 7-10.**
**The Maximum and Minimum Sizes for Icon Images**

| | Maximum Size | Minimum Size |
|---|---|---|
| **Default** | 50×50 pixels | 32×32 pixels |
| **Allowable** | 128×128 pixels | 16×16 pixels |

How the window manager treats an icon depends on the size of the image in relation to the maximum and minimum sizes.

**Table 7-11. Image Size Affects Icon Treatment**

| If an icon image is ... | The window manager will ... |
|---|---|
| Smaller than the minimum size. | Act as if you specified no image. |
| Within maximum and minimum limits. | Center the image within the maximum area. |
| Larger than the maximum size. | Clip the right side and bottom of the image to fit the maximum size. |

You can use the following two resources to control icon image size:

**Table 7-12. Controlling Icon Image Size**

| To specify this ... | Use this resource ... |
|---|---|
| Maximum size of an icon image. | `iconImageMaximum` |
| Minimum size of an icon image. | `iconImageMinimum` |

Bear in mind that the overall width of an icon is the image width *plus* border padding and the image height is the icon height *plus* border padding.

## Using Custom Pixmaps

When you iconify a client, either the client supplies its own icon image, the window manager supplies a default image, or you supply an image of your own.

There are two resources that tell the window manager where custom icons are located:

- The `iconImage` resource specifies the bitmap for a particular icon image. Its value is the path to the file containing the bitmap. If this resource is specified, it overrides any client-specified images.
- The `bitmapDirectory` resource causes the window manager to search a specified directory for bitmaps. The `bitmapDirectory` resource causes the window manager to search the specified directory whenever a bitmap is named with no complete path. The default value for `bitmapDirectory` is `/usr/include/X11/bitmaps`.

## Coloring and Tiling Icons

A number of resources enable you to specify the colors of icon elements.

**Table 7-13. Coloring and Tiling Icon Resources**

| To color this ... | Use this resource ... |
|---|---|
| Icon image background. | `iconImageBackground` |
| Left and upper bevel of icon image. | `iconImageTopShadowColor` |
| Right and lower bevel of icon image. | `iconImageBottomShadowColor` |
| Icon image foreground. | `iconImageForeground` |
| Right and lower bevels of an icon image. | `iconImageBottomShadowPixmap` |
| Left and upper bevels of an icon image. | `iconImageTopShadowPixmap` |

Default values for these resources are the icon's bottom and top shadow pixmaps specified using the `bottomShadowPixmap` and `topShandowPixmap` resources set by the entries `Mwm*icon*`*resource* or `Mwm*`*resource*.

## Using the Icon Box to Hold Icons

The icon box allows you to use an icon box to contain icons, rather than having stand-alone icons on the workspace.

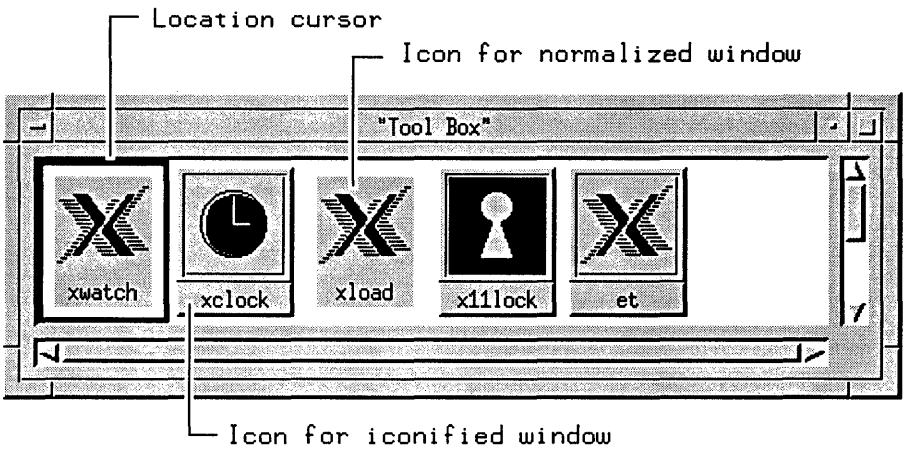


**Figure 7-4. Icon Box**

The icon box is a scrollable window that displays icons in a grid (rows and columns). Icons in the icon box do not overlap. If there are icons that cannot be displayed in the visible part of the icon box, you can scroll to see the icons. The sliders within the scroll bars show the extent of the icon grid that is visible.

The icon box can be minimized (iconified) just like any other window. If the icon box is minimized, it is placed into the icon grid on the workspace.

## Specifying the Icon Box

Several resources specify whether an icon box is used, define its geometry and location, and specify its name (for looking up resources) and title.

- The `useIconBox` resource specifies whether or not an icon box is used. A value of of "True" places icons in an icon box. The default value of "False" places icons on the root window.
- The `iconBoxGeometry` resource sets the initial size and placement of the icon box. If the `iconBoxGeometry` resource is used, the largest dimension of the size determines if the icons are placed in a row or a column. The default policy is to place icons in a row going from left to right, top to bottom.

The value of the `iconBoxGeometry` resource is a standard window geometry string with the following syntax:

- $Width \times Height \left[\pm x \pm y\right]$

If $x$ and $y$ are not provided, the icon box is placed at +0−0.

The actual size of the icon box window depends on the `iconImageMaximum` (size) and `iconDecoration` resources. The default value for size is (6 * iconWidth + padding) wide by (1 * iconHeight + padding) high.

- The `iconBoxName` resource specifies the name that is used to look up icon box resources. The default name is "iconbox."
- The `iconBoxTitle` resource specifies the name that is used in the title area of the icon box frame. The default name is "Icons."

For example, the following line specifies that icons will be placed in an icon box:

```
Mwm*useIconBox:  True
```

## Controlling the Appearance of Icon Boxes

The icon box is displayed in a standard window management client frame. Client-specific resources for the icon box can be specified using "iconbox" as the client name.

```
Mwm*iconbox*resource: value
```

Resources that can be used with the icon box to change its appearance are:

- `iconDecoration`.
- The `mwm` resources dealing with mattes and icon appearance. (The icon appearance resources affect the icon displayed when the icon box is minimized.)

## The Icon Box Window Menu

The window menu for the icon box differs from the standard window menu in that it does not contain the "Close" selection. In its place is the "PackIcons" selection, which shifts icons to fill empty spaces in the icon placement grid so that the icons appear in neat, complete rows.

## Controlling Icons in the Icon Box

Every client window that can be iconified has an icon in the icon box, even when the window is in the normal state. The icon for a client is put into the icon box when the client becomes managed by the window manager, and is removed from the icon box when the client stops.

Icons for windows in the normal (open) state are visually distinct from icons for windows that are iconified. Icons for windows that are iconified look like stand-alone icons. Icons for windows that are in the normal state appear flat and are optionally grayed-out. The value of "True" for the `fadeNormalIcon` resource grays out icons for normalized windows. The default value is "False."

The text and image attributes of icons in icon boxes are determined in the same way as for stand-alone icons, using the `iconDecoration` resource.

A standard "control" location cursor is used to indicate the particular icon in the icon box to which keyboard actions apply. The location cursor is an unfilled rectangle that surrounds the icon.

Icons contained in the icon box can be manipulated with the mouse and from the keyboard. Mouse button actions apply whenever the pointer is on any part of the icon.

**Table 7-14. Controlling Icons in the Icon Box With a Mouse**

| If you want to ... | Do this ... |
|---|---|
| Select an icon. | Press button 1. |
| Normalize (open) an iconified window. | Double-click mouse button 1. |
| Raise a normalized window to the top of the stack. | Double-click mouse button 1. |
| Move an icon within the icon box. | Drag button 1. |

To manipulate an icon from the keyboard, make the icon box the active window and use the arrow keys to traverse the icons in the icon box. Pressing (Return) does the default action for the selected icon: for an icon of a normalized window, the window is raised; for an icon of an iconified window, the window is normalized. The arrow keys move the focus around the icons that are visible. The (Tab) key moves the keyboard input focus around the box in this order: icons, horizontal scroll bar, vertical scroll bar, icons. (Shift) (Tab) moves the focus in the opposite direction.

## Managing Window Manager Menus

The OSF/Motif Window Manager menus are defined by a text file in your system directory called `system.mwmrc`, unless you have a file in your home directory called `.mwmrc`. You can add or delete menus and menu selections by copying `system.mwmrc` to your home directory as `.mwmrc` and modifying it to suit your needs.

### Default Menus

The OSF/Motif Window Manager comes with two default menus:

#### Default Window Menu

The default window menu is built into `mwm`. For reference, a copy of its contents are placed in `.mwmrc`.

```
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
}
```

7

By default, the window menu displays when you do the following operations:

- Press button 1 on a window frame's window menu button.
- Press button 3 anywhere on a window frame.
- Press Shift Esc with the keyboard focus set to a window.

The `windowMenu` resource must be set in order to replace the `DefaultWindowMenu` with a different menu.

## Default Root Menu

The default root menu is specified in the same files by the following lines:

```
Menu RootMenu
{
 "Root Menu"    f.title
 "New Window"   f.exec "hpterm &"
 "Start Clock"  f.exec "xclock -geometry 100x90-1+1 &"
 "Start Load"   f.exec "xload -geometry 150x90-130+1 &"
 "Shuffle Up"   f.circle_up
 "Shuffle Down" f.circle_down
 "Refresh"      f.refresh
  no-label      f.separator
 "Restart..."   f.restart
}
```

By default, the root menu displays when you press button 3 on the root window.

## Modifying Menus

You can modify either menu to suit the specific needs of your application; however, for the sake of the consistency of window operation, it's usually better to modify the root menu and keep the window menu the same.

All window manager menus, regardless of the mechanism that calls them to the screen, have the same syntax.

### Menu Syntax

```
Menu MenuName
{
   selection1 [mnemonic]  [accelerator]  function [argument]
   selection2 [mnemonic]  [accelerator]  function [argument]
   selection3 [mnemonic]  [accelerator]  function [argument]
                                     :
   selection* [mnemonic]  [accelerator]  function [argument]
}
```

Each line identifies a selection name followed by the function to be done if that selection is chosen. The order of the selections is the order of their appearance when you display the menu. A selection name may be either a character string or a bitmap.

## Selections

Any character string containing a space must be enclosed in double quotes (""); single-word strings don't have to be enclosed, but it's probably a good idea for the sake of consistency. An alternate method of dealing with two-word selection names is to use an underbar (_) in place of the space.

## Menu Titles

The `f.title` functions creates a menu title, and automatically places a separator above and below the title.

## Mnemonics and Accelerators

You have the option of using a mnemonic and accelerator with a menu selection. A mnemonic is specified using the syntax:

`mnemonic =` _*character*

An accelerator is specified using keyboard binding syntax described later in this chapter (see "Keyboard Binding Syntax").

## Functions

Each function operates in one or more of the following contexts:

| | |
|---|---|
| root | Operates the function when the root window is selected. |
| icon | Operates the function when an icon is selected. |
| window | Operates the function when a client window is selected. |

Additionally each function is triggered by one or more of the following devices:

7

| | |
|---|---|
| Button | Button binding (mouse). |
| Key | Key binding. |
| Menu | Window manager menu. |

Any selection that uses an invalid context, an invalid function, or a function that doesn't apply to the current context is grayed out. This is the case with the "Restore" selection of a terminal window's system menu or the "Minimize" selection of an icon's window menu.

The following table lists the valid functions, contexts, and devices.

Table 7-15. Valid Window Manager Functions

| Functions | | Contexts | | | Devices | | |
|---|---|---|---|---|---|---|---|
| Name | Description | Root | Icon | Window | Button | Key | Menu |
| f.beep | Causes a beep to sound. | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| f.circle_down | Puts window on bottom of stack. | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| f.circle_up | Puts window on top of stack. | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| f.exec | Uses `/bin/sh` to execute a command. | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| f.focus_color | Sets colormap focus when colormap focus policy is explicit. | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| f.focus_key | Sets keyboard input focus when keyboard focus policy is explicit. | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| f.kill | Terminates a client's connection to server. | | ✓ | ✓ | ✓ | ✓ | ✓ |
| f.lower | Lowers a window to bottom of stack. | | ✓ | ✓ | ✓ | ✓ | ✓ |
| f.maximize | Enlarges a window to its maximum size. | | ✓ | ✓ | ✓ | ✓ | ✓ |
| f.menu | Associates a menu with a selection or binding. | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| f.minimize | Changes a window into an icon. | | | ✓ | ✓ | ✓ | ✓ |
| f.move | Enables the interactive moving of a window. | | ✓ | ✓ | ✓ | ✓ | ✓ |

**Table 7-15a. Valid Window Manager Functions (continued)**

| Functions | | Contexts | | | Devices | | |
|---|---|---|---|---|---|---|---|
| **Name** | **Description** | **Root** | **Icon** | **Window** | **Button** | **Key** | **Menu** |
| f.next_cmap | Installs the next colormap in the window with the colormap focus. | √ | √ | √ | √ | √ | √ |
| f.next_key | Sets keyboard focus policy to the next window/icon in the stack. | √ | √ | √ | √ | √ | √ |
| f.nop | Does no function. | √ | √ | √ | √ | √ | √ |
| f.normalize | Displays a window in normal size. | | √ | √ | √ | √ | √ |
| f.pack_icons | Packs icons rows in the root window or icon box. | √ | √ | √ | √ | √ | √ |
| f.pass_keys | Toggles between enabling and disabling processing of key bindings. | √ | √ | √ | √ | √ | √ |
| f.post_wmenu | Posts the window menu | √ | √ | √ | √ | √ | |
| f.prev_cmap | Installs the previous color map in the window with the colormap focus. | √ | √ | √ | √ | √ | √ |
| f.prev_key | Sets the keyboard input focus to the next window/icon in the stack. | √ | √ | √ | √ | √ | √ |

**Table 7-15b. Valid Window Manager Functions (continued)**

| Functions | | Contexts | | | Devices | | |
|---|---|---|---|---|---|---|---|
| **Name** | **Description** | **Root** | **Icon** | **Window** | **Button** | **Key** | **Menu** |
| f.quit_mwm | Terminates OSF/Motif Window Manager, but not X. | √ | | | √ | √ | √ |
| f.raise | Lifts a window to the top of the window stack. | | √ | √ | √ | √ | √ |
| f.raise_lower | Raises a partially concealed window; lowers an unconcealed window. | | √ | √ | √ | √ | √ |
| f.refresh | Redraws all windows. | √ | √ | √ | √ | √ | √ |
| f.refresh_win | Redraws a client window. | | | √ | √ | √ | √ |
| f.resize | Enables you to interactively resize a window. | | | √ | √ | √ | √ |
| f.restart | Restarts the OSF/Motif Window Manager. | √ | | | √ | √ | √ |
| f.send_msg | Sends a client message. | | √ | √ | √ | √ | √ |
| f.separator | Draws a line between menu selections. | √ | √ | √ | | | √ |
| f.set_behavior | Restarts `mwm` with CXI or custom behavior. | √ | √ | √ | √ | √ | √ |
| f.title | Inserts a title into a menu at the specified position. | √ | √ | √ | | | √ |

## Changing the Menu Associated with the Window Menu Button

The `windowMenu` resource lets you change the *menu* displayed when you press button 1 on the window menu button.

For example, you would place the following line in your personal resource specifications to associate a menu named `EditMenu` with an `hpterm` window started as `hpterm -name hp850`.

```
    Mwm*hp850*windowMenu:    EditMenu
```

## Mouse Button Bindings

The window manager recognizes the following button operations:

| | |
|---|---|
| Press | Holding down a mouse button. |
| Click | Pressing and releasing a mouse button. |
| Double-click | Pressing and releasing a mouse button twice in rapid succession. |
| Drag | Pressing a mouse button and moving the pointer (and mouse device). |

You associate a button operation with a window management function using a **button binding**. A button binding is a command line you put in the `.mwmrc` file that associates a button operation with a window manager function.

### Default Button Bindings

The OSF/Motif Window Manager comes with the following built-in button bindings.

Table 7-16. Built-In Button Bindings

| Location of Pointer | Behavior |
|---|---|
| Window menu button | Pressing button 1 displays the window menu. This behavior can be modified by the `wMenuButtonClick` resource. |
| Window menu button | Double-clicking button 1 closes the window. This behavior can be modified by the `wMenuButtonClick2` resource. |
| Minimize button | Clicking button 1 minimizes the window. |
| Maximize button | Clicking button 1 maximizes the window. |
| Title bar | Dragging button 1 moves the window. |
| Window or icon | Pressing button 1 gives it keyboard focus. |
| Resize border | Dragging button 1 resizes the window. |
| Icon | Clicking button 1 displays the icon window menu. This behavior can be modified by the `iconClick` resource. |
| Icon | Double-clicking button 1 normalizes the window. |
| Icon | Pressing button 1 moves the icon. |

These bindings are fixed—they cannot be *replaced* by other bindings. However, you can *add* to some of them (see "Modifying Button Bindings and Their Functions.") For example, you can specify an additional function for double-clicking button 1 in an icon, but the double click will also normalize the window.

7

`Mwm` provides an additional default binding that can be deleted or replaced:

Table 7-17. Additional Button Bindings

| Locaton of Pointer | Behavior |
|---|---|
| Icon or Frame | Pressing button 1 raises the window or icon. |

This binding is listed in the following section of the `.mwmrc` file.

```
Buttons DefaultButtonBindings
{
 <Btn1Down>  icon|frame  f.raise
}
```

The binding can be removed or altered by deleting or editing the line that begins with `<Btn1Down>`. (In order for the editing to have an effect, the `buttonBindings` resource must be set to `DefaultButtonBindings`, and you must restart the window manager.)

## Modifying Button Bindings and Their Functions

You can modify the button bindings section of your `.mwmrc` file to suit your individual needs.

### Button Binding Syntax

The syntax for button bindings is as follows:

```
Buttons ButtonBindingSetName
{
   button  context|context  function  [argument]
   button  context|context  function  [argument]
   button  context|context  function  [argument]
}
```

The following button binding contexts are recognized by the window manager:

| | |
|---|---|
| `root` | Operates the function when the button is activated in the root window. |
| `window` | Operates the function when the button is activated in a client window or window frame. |
| `frame` | Operates the function when the button is activated on a window frame. |
| `icon` | Operates the function when the button is activated on an icon. |
| `title` | Operates the function when the button is activated on a title bar. |
| `app` | Operates the function when the button is activated in a client window (excludes window frames). |

### Modifying Button Bindings

Button bindings can be modified by:

- Editing the `DefaultButtonBindings` section of `.mwmrc`.
- Making a new button binding set.

To create a new button binding set:

1. Edit the `.mwmrc` file to include a new key binding set with a unique name.
2. Set the `buttonBindings` resource in your `.Xdefaults` file to the new name.

### Modifying Button Click Timing

The OSF/Motif Window Manager has another resource for controlling button behavior. This resource, `doubleClickTime`, sets the maximum time (in milliseconds) that can elapse between button clicks before a double-click becomes just "two clicks in a row." In other words, if two clicks occur in less than the maximum time, they are assumed to be a double-click; if two clicks occur in a time greater than the maximum time, they are assumed to be two single clicks. The default is 500 (milliseconds).

## Keyboard Bindings

Similar to mouse button bindings, you can bind (associate) window manager functions to "special" keys on the keyboard using **keyboard bindings**. The window manager recognizes the following special keys:

- Shift.
- Escape.
- Alt (Meta or Extend Char).
- Tab.
- Ctrl.
- Lock.

## Default Key Bindings

The OSF/Motif Window Manager comes with the following default key bindings.

**Table 7-18.**
**OSF/Motif Window Manager Default Keyboard Bindings**

| When the keyboard focus is: | Press: | What this does is: |
|---|---|---|
| Window or icon | Shift Escape | Displays window menu. |
| Window or icon | Alt space | Displays window menu. |
| Window, icon, or none | Alt Tab | Switches keyboard focus to the next window or icon. |
| Window, icon, or none | Alt Shift Tab | Switches keyboard focus to the previous window or icon. |
| Window, icon, or none | Alt Escape | Switches keyboard focus to the next window or icon. |
| Window, icon, or none | Alt Shift Escape | Switches keyboard focus to the previous window or icon. |
| Window | Alt F6 | Switches keyboard focus to the next window or icon, including transient windows. |
| Window, icon, or none | Alt Ctrl Shift ! | Restart `mwm` with default or custom behavior. |

These keyboard bindings are listed in the following lines in `system.mwmrc` and `.mwmrc`.

```
Keys DefaultKeyBindings
{
    Shift<Key>Escape          window              f.post_wmenu
    Alt<Key>space             window|icon         f.post_wmenu
    Alt<Key>Tab               root|icon|window    f.next_key
    Alt Shift<Key>Tab         root|icon|window    f.prev_key
    Alt<Key>Escape            root|icon|window    f.next_key
    Alt Shift<Key>Escape      root|icon|window    f.prev.key
    Alt<Key>F6                window              f.next_key transient
    Alt Ctrl Shift<Key>exclam root|icon|window    f.set_behavior
}
```

You can modify or delete any of these bindings, except "`Alt Ctrl Shift<Key>exclam`", by editing or deleting the line. (In order for the editing to have an effect, the `keyBindings` resource in the `.Xdefaults` file must be set to `DefaultKeyBindings`.)

## Modifying Keyboard Bindings and Their Functions

You can modify the keyboard bindings section of your `.mwmrc` file if your situation requires it.

### Keyboard Binding Syntax

The syntax for keyboard bindings is as follows:

```
Keys KeyBindingSetName
{
    key  context|context  function   [argument]
    key  context|context  function   [argument]
    key  context|context  function   [argument]
}
```

7

The following keyboard binding contexts are recognized by the window manager:

| | |
|---|---|
| root | Operates the function when the key is pressed while the root window has keyboard focus. |
| window | Operates the function when the key is pressed while a client window has keyboard focus. |
| icon | Operates the function when the key is pressed while an icon has keyboard focus. |

### Modifying Keyboard Bindings

Key bindings can be modified by:

- Editing the `DefaultKeyBindings` section in `.mwmrc`.
- Making a new key binding set.

To create a new keyboard binding set:

1. Edit `.mwmrc` to include the new key binding set with a unique name.
2. Set the `keyBindings` resource in your `.Xdefaults` file to the new name.

## Controlling Window Size and Placement

The following table lists window manager resources enabling you to refine your control over the size and placement of windows.

**Table 7-19.**
**Refining Your Control with Window Manager Resources**

| To control this ... | Use this resource ... | The default is ... |
|---|---|---|
| Initial placement of new windows on the screen. | `interactivePlacement` | False |
| The ability to enlarge windows beyond the size specified in `maximumClientSize`. | `limitResize` | False |
| The maximum size of a client window set by user or client. | `maximumMaximumSize` | 2×screen |
| The sensitivity of dragging operations. | `moveThreshold` | 4 pixels |
| Exact positioning of window and window frame. | `positionIsFrame` | True |
| Clipping of new windows by screen edges. | `positionOnScreen` | True |
| The width of the resize border of the window frame. | `resizeBorderWidth` | 10 pixels |
| Displaying the resize cursors when the pointer is in the resize border. | `resizeCursors` | True |
| The maximum size of a maximized client. | `maximumClientSize` | screen size |

The `interactivePlacement` resource has the following values:

| | |
|---|---|
| True | The pointer changes shape (to an upper left corner bracket) before a new window displays, so you can choose a position for the window. |
| False | The pointer doesn't change shape. A new window displays according to the placement values specified in the X configuration files. |

The `limitResize` resource has the following values:

| | |
|---|---|
| True | A window cannot be resized to greater than the maximum size specified by the maximumClientSize resource or the WM_NORMAL_HINTS window property. |
| False | A window can be resized to any size. |

The value of the maximumMaximumSize resource is the width×height of the screen being used. The dimensions are given in pixels. For example, for an SRX display, maximumMaximumSize would have a value of 1280×1024.

The value of the moveThreshold resource is the number of pixels that the pointer must be moved with a button pressed before a move operation is initiated. You can use this resource to prevent window or icon movement when you unintentionally move the pointer during a click or double-click.

The positionIsFrame resource has the following values:

| | |
|---|---|
| True | The position information (from WM_NORMAL_HINTS and configuration files) refers to the position of the window frame. |
| False | The position information refers to the position of the window itself. |

The positionOnScreen resource has the following values:

| | |
|---|---|
| True | If possible, a window is placed so that it is not clipped. If not possible, a window is placed so that at least the upper left corner of the window is on the screen. |
| False | A window is placed at the requested position even if it is totally off the screen. |

The value of the resizeBorderWidth resource is the width of the resize border, the outermost portion of the window frame. The width is measured in pixels.

The resizeCursors resource has the following values:

| | |
|---|---|
| True | The appropriate resize cursor displays when the pointer enters a resize border area of the window frame. |
| False | The resize cursors are not displayed. |

The value of the maximumClientSize resource is the width×height (in pixels) of the maximum size of a maximized client. If this resource isn't specified, the maximum size is taken from the WM_NORMAL_HINTS window property, or the default size (the size of the screen) is used.

For example, you might decide that xload clients should be maximized to no more than an eighth of the size of your 1024×768 display.

```
Mwm*XLoad.maximumClientSize: 128x96
```

## Controlling Focus Policies

The focus policies determine what happens when a window becomes the active window. The active window is the window that has the focus of the keyboard and any extended input devices. When a window is active, the following are true:

- What you type appears in that window.
- The color of the window frame changes to indicate the active focus.
- Input from extended input devices goes to that window.

Each focus policy is controlled by a specific focus policy resource. The focus policy resources are as follows:

**Table 7-20.
Controlling Focus Policies with Window Manager Resources**

| To control this ... | Use this resource ... | The default value is ... |
|---|---|---|
| Which client window has the colormap focus. | `colormapFocusPolicy` | keyboard |
| Which client window has the keyboard and mouse focus. | `keyboardFocusPolicy` | explicit |

7

The following focus policies are valid for the `colormapFocusPolicy` resource:

| | |
|---|---|
| keyboard | The window manager tracks keyboard input and installs a client's colormap when the client window gets the keyboard input focus. |
| pointer | The window manager tracks the pointer and installs a client's colormap when the pointer moves into the client window or the window frame around the client. |
| explicit | The window manager tracks a specific focus-selection operation and installs a client's color map when the focus-selection operation is done in the client window. |

The following focus policies are valid for the `keyboardFocusPolicy` resource:

| | |
|---|---|
| pointer | The window manager tracks the pointer and sets the keyboard focus to a client window when the pointer moves into that window or the window frame around the client. |
| explicit | The window manager tracks a specific focus-selection operation and sets the keyboard focus to a client window when the focus-selection operation is done in that client window. |

When the keyboard focus policy is explicit, you can use the `passSelectButton` resource to specify the consequence of the focus-selection operation. If you give `passSelectButton` a value of "True" (the default value), the focus-selection operation is passed to the client or used by the window manager to perform some action. If you give `passSelectButton` a value of "False," the focus-selection operation will be used only to select the focus and will not be passed.

7

For example, you could change the keyboard focus policy so that moving the pointer into a window moved the focus there by adding the following line in your `.Xdefaults` file:

```
Mwm*keyboardFocusPolicy: pointer
```

## Switching Between Default and Custom Behavior

The window manager has a built-in key binding that allows you to switch back and forth between customized mwm behavior and default behavior. The key presses for doing this are Alt Shift Ctrl !.

The following client-specific resources are affected by this function:

```
clientDecoration    clientFunctions    focusAutoRaise    windowMenu
```

## Using the Window Manager with Multiple Screens

By default, the mwm manages one screen. Managing multiple screens can be specified in two ways:

- Using resources.
- Editing the startup command for mwm.

### Using Resources to Manage Multiple Screens

The following resources configure the window manager to manage multiple screens:

- To specify that mwm manage multiple screens, use the resource:

  ```
  Mwm*multiScreen:   True
  ```

  This tells mwm to try to manage all screens that the server manages.

- To define the screen names, use the resource screenList. For example, the following resource names two screens zero and one.

  ```
  Mwm*screenList:  zero  one
  ```

### Specifying Multiple Screens from the Command Line

You can use command-line options to start mwm so that it manages multiple screens.

- The -display option specifies the display. It has the syntax:

  -display *hostname*:*display*.*screen*

- The `-multiscreen` option causes `mwm` to manage all the screens on the specified display.
- The `-screens` option specifies the screen names used to obtain screen-specific resources.

For example,

```
mwm -display local:0.1 -multiscreen -screens zero one
```

causes `mwm` to manage all the screens on display 0. Screens 0 and 1 are named `zero` and `one`.

# Using the X Clients

Programs running in the X environment can be divided into two groups:

| | |
|---|---|
| **X clients** | "Window-smart" programs written for the X Window System. |
| **non-clients** | Programs written for terminals. Non-clients are run in terminal emulation windows. |

Related chapter:

- Chapter 5 covers setting resources for clients, client option, and the display, and geometry resources.

## Starting Clients and Non-clients

Programs can run as either background or foreground processes. In any X11 terminal window, you can run only one program as a foreground process, but you can run many programs as background processes. To run a program as a background process, add an ampersand (&) to the end of the command line that starts the program.

The general syntax for the command line that starts a client is:

*client* [*-options*] [&]

An & at the end of the command line causes the client to start as a background process.

Programs can be started:

- From the command line.

  The client name and options are typed after the command line prompt.

- From menus.

  Refer to chapter 7 for details of how to create your own menus.

- As part of the X startup.

  Refer to chapter 4 for information about the `.x11start` file.

## Command-Line Options

Command-line options override all default files. If no options are specified, the client is started using resource values from the resource database, the client's app-defaults, or from defaults built into the client.

Some **toolkit options** are common to most clients:

| | |
|---|---|
| `-fn` *font* | Specifies the font to use for the client. |
| `-bg` *color* | Specifies the background color. |
| `-fg` *color* | Specifies the foreground color. |
| `-display` *host:display.screen* | Specifies the host where the client will display its output. |
| `-geometry` *width×height* | Specifies the size of the window and its location. |
| `-help` | Displays an explanation of the options available for the client. |

For a specific client's options, refer to the client's man page in the Reference Section.

Options have the syntax:

8

*-option argument*

For example, the following command line starts an `hpterm` window with a black background and white foreground:

```
hpterm  -bg  Black  -fg  White &
```

## Specifying the Display and Screen

The default display on which a client is displayed is obtained from the DISPLAY environment variable of the system on which the client starts. It sets the host, display number, and screen number to which the client directs its output. This is typically display 0, screen 0.

Most clients have a –display option that lets you set the host, display number, and screen on which the client will display its output. The -display option has the syntax:

```
–display [host:display.screen]
```

| | |
|---|---|
| *host* | The hostname of a valid system on the network. |
| *display* | The number of the display on the system on which you want the output to appear. A display can include more than one screen. |
| *screen* | The number of the screen where the output is to appear. The default is 0. |

For example, executing the command:

```
hpterm  –display  hpxhere:0.1 &
```

starts an hpterm process on the local system and displays the window on display 0, screen 1 of the hpxhere system. The window has the default size, location, and color.

## Starting Remote Programs

A remote client is a client that runs on a computer other than the computer running the X server. In other words, a remote client runs on one computer while its output is displayed on another.

There are several ways to run programs on a remote host from a command line:

- Use `rlogin` to log into the remote host.
- Use `remsh` to start a client remotely without formally logging in.

If the client produces output on a display, you must specify the display and screen on which you want the output to appear.

### Running Programs Using 'rlogin'

You can use an existing terminal emulator window to log into a remote host. Once the window is acting as a terminal off the remote host, you can run clients there and direct the output to any display.

For example, the following commands log into and start `xload` on remote host `hpthere` and display the output on local system `hpxhere`.

```
rlogin  hpthere
xload  -display  hpxhere:0.0 &
```

### Using 'remsh' to Start Programs

The benefit of using `remsh` instead of `rlogin` is that the the local system starts only one process (the client) with a remote shell; with the remote login, the local system starts both the remote login and the client.

#### Starting Clients Remotely

The following syntax starts a remote shell on a remote host, redirects `remsh` input, starts a client, and directs output to the local display.

8

`remsh` *remote* `-n` *client* `-display` *local*`:`*display*`.`*screen* `&`

| | |
|---|---|
| *remote* | The remote host name. |
| *client* | Absolute path of the executable client file (`remsh` does not allow the `PATH` variable). |
| *local* | Local host name. |

For example, the following command runs `xload` on remote host `hpthere` and directs output to the display of system `hpxhere`.

```
   remsh hpthere -n /usr/bin/X11/xload -display hpxhere:0.0 &
```

Generally, `remsh` is preferred to `rlogin` for starting a remote program from a menu. For example, the following line added to the workspace menu starts a remote `hpterm` window on remote host `hpthere`:

```
"Doc files"  f.exec "remsh hpthere -n /usr/bin/X11/hpterm -display hpxhere:0.0 &"
```

### Starting a Remote Non-Client

At the command-line prompt of an existing window, you could execute:

```
   hpterm -display hpxhere:0.0 -e remsh hpthere -n ll &
```

This example starts a new `hpterm` client and directs its output to the local display (`-display hpxhere`). The `-e` option executes a remote shell on `hpthere` that connects the window to the remote host `hpthere` and lists the files in your home directory there. When the `ll` command finishes executing, the window created for it to run in will disappear. Thus, this method of starting remote non-clients is usually not desirable.

## Stopping Programs

If a program has data you want to save, you must save the data *before* you stop it.

If a terminal window is running a non-client containing data, you must stop the non-client in the approved manner before you stop the window. Generally, a non-client has a "stop" provision, or stops when it has finished executing.

After you have saved any data and exited any non-clients (in the case of terminal windows), stop the client by choosing the "Close" selection from the client's window menu.

Note that if you started a non-client as an option of creating a window, when you stop the non-client, the window will stop.

If you are unable to stop a program in the normal manner, you should "kill" the program before you log out.

To kill a program, first try these keystrokes:

- Press CTRL c.
- Press CTRL d.
- Press q.
- Press ESC, then :, then q.

If these don't work, use the HP-UX `kill` command to stop the program's execution environment or "process." To use the `kill` command:

1. Save any data that needs saving.
2. Find the PID (process ID) by executing:

   ```
   ps -fu login_name
   ```

3. To kill the program, execute:

   ```
   kill -2 pid
   ```

   where *pid* is the PID number. This is equivalent to CTRL c.
4. If this doesn't work, execute:

   ```
   kill -3 pid
   ```

5. If this still doesn't work, execute:

   ```
   kill -9 pid
   ```

Certain programs are *cached* during a session; that is, once they are started, closing them unmaps the window but does not stop the process. If you need to halt one of these processes during a session, use the `kill` command.

## The X Clients

The following tables list the X clients.

**Table 8-1. Initialization and Configuration Clients.**

| Client | Description | Covered in Chapter ... |
|---|---|---|
| `xmodmap` | Alters the modifier-key mappings of a keyboard. | 9 |
| `xset` | Adjusts display preference options for a session. | 8 |
| `xinitcolormap` | Initializes a new colormap for an X environment. | 8 |
| `rgb` | Creates a color database for X. | 8 |
| `xhost` | Adds a new remote host to your system. | 8 |
| `xrdb` | Loads a window manager's resource configuration into the server. | 5 |
| `xinit` | Starts the X server and clients. | 4 |
| `x11start` | Starts the X11 Window System using `xinit`. | 4 |

**Table 8-2. Window Management Clients.**

| Client | Description | Covered in Chapter ... |
|---|---|---|
| `resize` | Sets the environment to reflect the correct window size. | 8 |
| `xwininfo` | Displays information about windows. | 8 |

Table 8-3. Graphics Functions Clients.

| Client | Description | Covered in Chapter ... |
|---|---|---|
| `xseethru` | Opens a window into a graphics workstation image plane when the X Window System is running in the overlay planes. | 11 |
| `xwd` | Makes a pixmap screen dump in `xwd` format. | 10 |
| `xwd2sb` | Translates an `xwd` pixmap to Starbase format. | 11 |
| `sb2xwd` | Translates a Starbase pixmap to `xwd` format. | 11 |
| `xpr` | Prints a screen dump. | 10 |
| `gwind` | Creates a window for Starbase applications. | 11 |
| `gwindstop` | Stops multiple Starbase X windows. | 11 |
| `xwcreate` | Creates a new X window for Starbase. | 11 |
| `xwdestroy` | Destroys a Starbase X window. | 11 |
| `xwud` | Displays a previously made screen dump. | 10 |

Table 8-4. Viewable Services Clients.

| Client | Description | Covered in Chapter ... |
|---|---|---|
| `xterm` | Terminal emulator for a DEC or Tektronix terminal. | 8 |
| `hpterm` | Terminal emulator for Term0 terminals. | 8 |
| `xclock` | Displays an analog or digital clock. | 8 |
| `xload` | Displays the system load average. | 8 |
| `xsetroot` | Sets the color and appearance of the root window. | 8 |

Table 8-5. Font Management Clients

| Client | Description | Covered in Chapter ... |
|---|---|---|
| `bdftopcf` | Compiles a BDF-formatted font into an X server format. | 6 |
| `mkfontdir` | Creates a `fonts.dir` file. | 6 |
| `xlsfonts` | Lists the fonts that match a given pattern. | 6 |

The following clients do not require X to be running: `rgb`, `xpr`, `xwd2sb`, `sb2xwd`, and `mkfontdir`.

## Terminal Emulation Clients

The X Window System has two terminal emulation clients, hpterm, and xterm. The default for HP-UX is hpterm.

### The 'hpterm' Client

The hpterm client emulates a Term0 terminal.

The syntax of the hpterm client is as follows:

hpterm [-*options*] [&]

There are too many options to cover here. Refer to the hpterm man page for all the options available.

### The 'xterm' Client

The xterm client emulates DEC VT102 and Tektronix 4014 terminals.

The syntax of the xterm client is as follows:

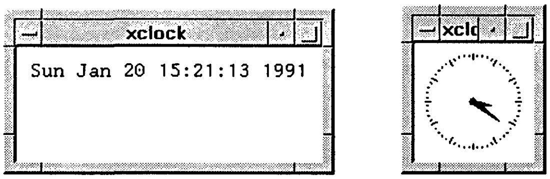
xterm [-*options*] [&]

There are too many options to cover here. Refer to the xterm man page for all the options available.

## The 'xclock' Client

The xclock client displays an analog or digital clock. The digital clock also displays the day, date, time, and year; the format automatically varies for local language custom based on the value of the LANG environment variable.

8



```
Sun Jan 20 15:21:13 1991
```

**Figure 8-1. Digital and Analog Clocks**

The syntax for the `xclock` client is:

```
xclock [-options] [&]
```

For a complete list of `xclock` options, refer to the `xclock` man page.

The following example creates a digital clock that updates every 10 seconds.

```
xclock -digital -update 10 &
```

The next example creates an analog clock that chimes every 30 minutes, updates every 5 seconds, and has yellow hands (all the other colors are the default ones).

```
xclock -analog -chime -update 5 -hd yellow &
```

## The 'xload' Client

The `xload` client displays a periodically updated histogram of the system load.

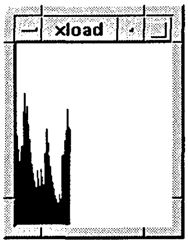


**Figure 8-2. The 'xload' Client**

The syntax for the `xload` client is:

```
xload [-options]
```

where:

| | |
|---|---|
| `-hl` *color* | The color of the scale lines. |
| `-jumpscroll` *number* | The number of pixels to shift the graph to the left when the graph reaches the right edge of the window. The default is half the width of the current window. |

| | |
|---|---|
| `-label` *string* | The string to put into the label above the histogram. |
| `-nolabel` | No label is displayed above the histogram. |
| `scale` *integer* | The number of tic marks in the histogram. The default is 1. |
| `update` *seconds* | The frequency at which the histogram is updated. |

`xload` also accepts the toolkit command line options.

## Customizing the Root Window with 'xsetroot'

The `xsetroot` client lets you:

- Customize the appearance of the root window.
- Change the bitmap used for the root window cursor.

The `xsetroot` client has the syntax:

```
           ┌-help                           ┐
           │-def                            │
           │-cursor path/cursor path/mask   │
           │-bitmap path/bitmap             │
           │-mod x y                        │
  xsetroot │-gray                           │
           │-fg color                       │
           │-bg color                       │
           │-rv                             │
           │-solid color                    │
           └-display host:display.screen    ┘
```

8

| | |
|---|---|
| `-help` | Prints a summary of the command usage. |
| `-def` | Resets unspecified root window attributes to their default values. |
| `-cursor` | Specifies the cursor bitmap and mask bitmap to use for the root window cursor. |
| `-bitmap` | Specifies a bitmap file with which to tile the root window. |

| | |
|---|---|
| `-mod` | Specifies a modular grid of dimensions $x$ by $y$ in the foreground color, making a plaid pattern. |
| `-gray` | Specifies gray (or grey) for the color of the root window. |
| `-fg` | Specifies *color* as the foreground color. |
| `-bg` | Specifies *color* as the background color. |
| `-rv` | Swaps foreground and background colors. |
| `-solid` | Specifies the root window should be colored a solid *color*. |
| `-display` | Specifies the host, display number, and screen number of the root window to change. |

For example, the following command changes the workspace cursor using two custom bitmaps located in directory `$HOME/bits`.

```
xsetroot -cursor ~/bits/shuttle.bm ~/bits/mask.bm
```

## Changing Display Preferences with 'xset'

The `xset` client allows you to change certain user preference options of the display. Note that hardware limitations and implementation differences may affect the results of the `xset` client.

`xset` provides a way to set:

- Bell volume, pitch, and duration.
- Keyboard click volume and autorepeat.
- Mouse acceleration and threshold.
- Font paths.
- Screen saver time.

The syntax for `xset` is:

```
      [ ⎧ -b                                              ⎫ ]
      [ ⎨ b ⎧ on                                      ⎫   ⎬ ]
      [ ⎪   ⎨ off                                     ⎬   ⎪ ]
      [ ⎩   ⎩ volume [ ,pitch, [ ,duration] ]         ⎭   ⎭ ]
      [ ⎧ -c                ⎫                              ]
      [ ⎨ c ⎧ on       ⎫    ⎬                              ]
      [ ⎪   ⎨ off      ⎬    ⎪                              ]
      [ ⎩   ⎩ [0-100]  ⎭    ⎭                              ]
      [ -fp path[ ,path...]                                ]
      [ fp- path[ ,path...]                                ]
      [ +fp path[ ,path...]                                ]
      [ fp+ path[ ,path...]                                ]
      [ fp ⎧ default              ⎫                        ]
      [    ⎩ path[ ,path...]      ⎭                        ]
      [ fp= path                                           ]
      [ fp rehash                                          ]
xset  [ m ⎧ acceleration threshold ⎫                       ]
      [   ⎩ default                ⎭                       ]
      [ p pixel color                                      ]
      [ pm ⎧ default ⎫                                     ]
      [    ⎩ number  ⎭                                     ]
      [ ⎧ -r              ⎫                                ]
      [ ⎨ r ⎧ on  ⎫       ⎬                                ]
      [ ⎩   ⎩ off ⎭       ⎭                                ]
      [ s ⎧ length period ⎫                                ]
      [   ⎪ blank         ⎪                                ]
      [   ⎪ noblank       ⎪                                ]
      [   ⎨ expose        ⎬                                ]
      [   ⎪ noexpose      ⎪                                ]
      [   ⎪ default       ⎪                                ]
      [   ⎪ on            ⎪                                ]
      [   ⎩ off           ⎭                                ]
      [ q                                                  ]
      [ -display host:display.screen                       ]
```

8

| | |
|---|---|
| `-b` | Turns the bell off. |

| | |
|---|---|
| `b on/off` | Turns the bell on or off. |
| `b v[p[d]]` | Specifies the bell volume, pitch, and duration. *Volume* is a percentage between 0 and 100 and can be specified without specifying pitch and duration. *Pitch* is in hertz and is specified together with a volume. *Duration* is in milliseconds and is specified with both volume and pitch. If only one parameter is given, it is taken as the volume. If two parameters are given, they are taken as volume and pitch. |
| `-c` | Turns the key click off. |
| `c on/off` | Turns the key click on or off. |
| `c` *0-100* | Specifies the key click volume as a percentage between 0 and 100. |
| `-fp/fp-` | Removes the specified directories from the font path. |
| `+fp/fp+` | Prefixes or appends the specified directories to the font path (depending on the position of the +). |
| `fp default` | Restores the default font path. |
| `fp` *path* | Specifies the font path, absolutely. |
| `fp=` *path* | Sets the font path. |
| `fp rehash` | Causes the server to reread the `fonts.dir` file and the `fonts.alias` files in each path of the server's font path. |
| `m` *acceleration threshold* | Specifies the acceleration and threshold of the mouse. *Acceleration* indicates the change in mouse speed (for example: 2=double, 3=triple). *Threshold* indicates the number of pixels of movement required before acceleration takes place. If only one number is given, it is taken as the acceleration. |
| `m  default` | Resets the mouse acceleration and threshold to their default values. |
| `p` *pixel color* | Controls color on a per pixel basis. *Pixel* is an integer representing a specific pixel in the X server's colormap. The exact number of pixels in the colormap depends on your hardware. *Color* specifies the color that pixel should be. |
| `pm default` | Restores the default font button codes to the pointer map. |
| `pm` *number* | Specifies the button codes for pointer map entries. |

| | |
|---|---|
| `-r` | Turns autorepeat off. |
| `r on/off` | Turns autorepeat on or off |
| `s` *length period* | Sets the screen saver option on. *Length* is the number of seconds that the server must be inactive before the screen is blanked. *Period* is the number of seconds a particular background pattern will be displayed before changing it. |
| `s blank` | Specifies that the screen saver should blank the video, if permitted by your hardware, rather than display the background pattern. |
| `s noblank` | Specifies that the screen saver should display the background pattern rather than blank the video. |
| `s expose` | Specifies that the server should discard window contents. |
| `s noexpose` | Specifies that the server should not enable the screen saver unless it saves window contents. |
| `s default` | Sets the system to its default screen saver characteristics. |
| `s on/off` | Sets the screen saver feature on or off. |
| `q` | Displays the current settings. |
| `-display` | Specifies the host, display number, and screen to be reset with `xset`. |

## Creating a Custom Color Database with 'rgb'

The `rgb.txt`, `rgb.pag`, and `rgb.dir` files in your system directory make up the color data base for the X Window System. It contains all the named colors and the amount of red, green, and blue needed to make the color. The following lines are from `rgb.txt`. Note that the red, green, and blue values are given as the decimal equivalents of their hexadecimal values.

**Table 8-6. Some Lines from 'rgb.txt'.**

| Red | Green | Blue | Color Name |
|---|---|---|---|
| 47 | 47 | 100 | MidnightBlue |
| 35 | 35 | 117 | navy blue |
| 35 | 35 | 117 | NavyBlue |
| 35 | 35 | 117 | navy |
| 35 | 35 | 117 | Navy |
| 114 | 159 | 255 | sky blue |
| 114 | 159 | 255 | SkyBlue |

As the above lines illustrate, several lines are sometimes necessary to account for alternative spellings of the same color.

Depending on your needs, you may want to make your own custom color database modeled after the `rgb.txt` file.

Hewlett-Packard recommends that your custom color database have a name other than `rgb.txt`. You can either copy `rgb.txt` and make your changes, or start with an entirely new file. In either case, the file entries are in the following format:

*redvalue greenvalue bluevalue name*

The fields are separated by either tabs or spaces.

The `rgb.txt` file is the source file used by the `rgb` client to make two other files that are used by the server: `rgb.dir` and `rgb.pag`. If you run `rgb` without any parameters, it will use `rgb.txt`. If you want to use your custom database, use the following syntax:

`rgb` *outfile <infile*

where *infile* is the name of your custom database, the text file you created. The `rgb` client will create *outfile*`.dir` and *outfile*`.pag`. 8

To put your new color database into effect, you must add it to your `.x11start` file. For example, if your new database is composed of the files `2brite.txt`, `2brite.dir`, and `2brite.pag` in the `/user/ellen` directory, type the following command line to start your X environment:

```
.x11start -- -co /user/ellen/2brite
```

The server assumes the color database is in the `/usr/lib/X11` directory unless told otherwise.

Note that recent X11 releases from Hewlett-Packard may contain more than one color database file, each customized for a particular display type for color consistency across display types. To avoid overwriting an existing `rgb.txt` file, the installation process for X11 does not automatically replace this file in `/usr/lib/X11`, but installs the new `rgb.txt*` file(s) in the directory `/etc/newconfig` or `/etc/newconfig/X11R*`. You must manually process (using the `rgb` utility) the desired `rgb.txt*` file in order to use one of the new versions. You should copy the desired `rgb.txt*`, `rgb.dir`, and `rgb.pag` files to the `/usr/lib/X11` directory. You may want to save the existing version of each file in `/usr/lib/X11` before copying the new version in.

## Initializing the Colormap with 'xinitcolormap'

The `xinitcolormap` client initializes the X colormap. Specific X colormap entries (pixel values) are made to correspond to specified colors. An initialized colormap is required by applications that assume a predefined colormap (for example, many applications that use Starbase graphics).

`xinitcolormap` reads a colormap file to determine the allocation of colors in the X colormap. The name of the colormap file is determined by using (in the following order):

1. The command line option [-f colormapfile].
2. `.Colormap` default value.
3. The `xcolormap` file in your system directory.
4. If no colormap file is not found, this default colormap specification is assumed— black (colormap entry 0), white, red yellow, green, cyan, blue, magenta (colormap entry 7).

`xinitcolormap` should be the first client program run at the start of a session in order to assure that colormap entries have the color associations specified in the colormap file. Sometimes you may encounter this X toolkit warning:

```
X Toolkit Warning: cannot allocate colormap entry for 94c4d0
```

where "94c4d0" is a color specified in the application running. If this occurs, it means that you have probably reached the limit of colors for your graphics card/display comnination. Executing `xinitcolormap` should solve the problem.

## Adding and Deleting Hosts with 'xhost'

Using `xhost`, you can add or delete a remote host's permission to access the local display server.

For example, the following command allows the remote host hpggggg to access your local display.

```
xhost  +hpggggg
```

## Resetting Environment Variables with 'resize'

This client is only available on HP-UX.

The `resize` client resets three environment variables: TERM, LINES, and COLUMNS. This enables a shell to reflect the current size of its window.

Don't confuse `resize`, the client, with `f.resize` the window manager function. The `f.resize` function changes the size of a window, but does not reset any environment variables. The `resize` client, on the other hand, does not change the size of a window, but it does reset the environment variables. Resetting the environment variables enables non-client programs to adjust their output to the window's new size.

Use `resize` whenever you resize a terminal emulator window and want a non-client program running in that window to reflect the window's new size. The `resize` client is typically used as an argument to the HP-UX `eval` command.

The syntax for `resize` is as follows:

```
         ⎧ -c            ⎫
         ⎪ -h            ⎪
resize   ⎨ -s [row col]  ⎬
         ⎪ -u            ⎪
         ⎩ -x            ⎭
```

| | |
|---|---|
| `-c` | Resets the environment variables for `csh` shells. |
| `-h` | Uses Hewlett-Packard terminal escape sequences to determine new window size. |
| `-s` | Uses Sun escape sequences to determine new window size. New row and column sizes are specified with *row* and *col.* |
| `-u` | Resets the environment variables for `sh` and `ksh` shells. |
| `-x` | Uses VT102 escape sequences to determine new window size. |

To see what the current COLUMN and LINES settings are, type the following command:

```
resize [Return]
```

After you have resized a window either by dragging the window frame or by choosing the "Size" selection from the window menu, you can reset the LINES, and COLUMN environment variables to reflect the new window size by issuing the following command:

```
eval ‘resize‘ Return
```

If you find yourself typing the above command too often, you can make things a little easier on yourself. If you use csh, try using an alias. The following line in your .cshrc file enables you to run resize by typing xr.

```
alias xr ’set noglob; eval ‘resize‘’
```

If you use sh or ksh create an xr function like the following:

```
xr()  {eval ‘resize‘;}
```

## Getting Window Information with ‘xwininfo’

The xwininfo client is a utility program that displays useful information about windows.

The syntax for xwininfo is as follows:

```
          [ -help                            ]
          [ { -id id       }                 ]
          [ { -name name   }                 ]
          [ { -root        }                 ]
          [ -int                             ]
          [ -tree                            ]
          [ -stats                           ]
xwininfo  [ { -metric  }                     ]
          [ { -english }                     ]
          [ -bits                            ]
          [ -events                          ]
          [ -size                            ]
          [ -wm                              ]
          [ -all                             ]
          [ -display host:display.screen     ]
```

| | |
|---|---|
| -help | Prints a summary of the command usage. |
| -id | Specifies the target window by window id. |

8

| | |
|---|---|
| `-name` | Specifies the target window by name. |
| `-root` | Specifies the root window as the target. |
| `-int` | Displays window information, normally shown as hexadecimal, as decimal. |
| `-tree` | Displays ids and names of the root, parent, and child windows. |
| `-stats` | Displays window id, location, size, depth, and other information as hexadecimal. |
| `-metric` | Displays height, width, x and y information in millimeters. |
| `-english` | Displays height, width, x and y information in inches, feet, yards. |
| `-bits` | Displays information about bit and storage attributes. |
| `-events` | Displays event masks of the target window. |
| `-size` | Displays sizing information about the target window. |
| `-wm` | Displays the window manager hints for the target window. |
| `-all` | Displays all available information about a window. |
| `-display` | Specifies the host, display, and screen to target. |

This example illustrates the result of issuing the following command:

```
xwininfo -stats Return
```

Once you issue the command, select a window as the target of your inquiry by moving the pointer into that window and clicking button 1.

```
xwininfo ==> Window id: 0x200013 (hpaaaaa)
  ==> Upper left X: 6
  ==> Upper left Y: 6
  ==> Width: 484
  ==> Height: 316
  ==> Depth: 8
  ==> Border width: 4
  ==> Window class: InputOutput
  ==> Colormap: 0x80065
  ==> Window Bit Gravity State: NorthWestGravity
  ==> Window Window Gravity State: NorthWestGravity
  ==> Window Backing Store State: NotUseful
  ==> Window Save Under State: no
==> Window Map State: IsViewable
==> Window Override Redirect State: no
==> Corners:  +6+6  -782+6  -782-694  +6-694
-geometry =80x24+6+6
```

9

# Customizing the Mouse and Keyboard

This chapter describes the following customizations:

- Changing mouse button actions.
- The `xmodmap` client.
- Going mouseless.
- Customizing keyboard input.

Related information:

- Chapter 7 contains `mwm` mouse and keyboard bindings.

## Changing Mouse Button Actions

Normally, the mouse pointer buttons are mapped as follows:

**Table 9-1. Default Mouse Button Mapping.**

| Button Number | Button on a 2-button mouse | Button on a 3-button Mouse |
|---|---|---|
| Button 1 | Left button | Left button |
| Button 2 | Both buttons simultaneously | Middle button |
| Button 3 | Right button | Right button |
| Button 4 | | Left and middle buttons simultaneously |
| Button 5 | | Middle and right buttons simultaneously |

However, you can change these mappings. To generate buttons 4 and 5 on a three-button mouse, you must enable button chording as described later in this chapter.

**Table 9-2. Alternative Mouse Button Mappings.**

| To press Button | Left Hand Mapping | | OSF/Motif Mapping | |
|---|---|---|---|---|
| | 2-button mouse | 3-button mouse | 2-button mouse | 3-button mouse |
| Button 1 | Right button | Right button | Left button | Left button |
| Button 2 | Both buttons simultaneously | Middle button | Right button | Middle button |
| Button 3 | Left button | Left button | Both buttons simultaneously | Right button |
| Button 4 | | Middle and right buttons simultaneously | | Left and middle buttons simultaneously |
| Button 5 | | Middle and left buttons simultaneously | | Right and middle buttons simultaneously |

The xmodmap utility can be used to change mouse button mappings. The syntax for changing mouse button mappings with xmodmap is:

```
xmodmap [ -e " [ pointer = default                ] " ]
        [      [ pointer = number[ number...]     ]   ]
        [ -pp                                         ]
```

| | |
|---|---|
| `-e` | Specifies a remapping expression. Valid expressions are covered in "Customizing Keyboard Input" later in this chapter. |
| `default` | Set mouse keys back to default bindings |
| `number` | Specifies a list of button numbers to map the mouse keys to. The order of the numbers refers to the original button mapping. |
| `pp` | Print the current pointer mapping. |

For example, to reverse the positions of buttons 1 and 3 for left-handed mapping:

```
xmodmap -e "pointer = 3 2 1"       2-button mouse
xmodmap -e "pointer = 3 2 1 5 4"   3-button mouse
```

To establish OSF/Motif-standard button mapping:

```
xmodmap -e "pointer = 1 3 2"       2-button mouse
xmodmap -e "pointer = 1 3 2 4 5"   3-button mouse
```

## Going Mouseless with the 'X*pointerkeys' File

Your work situation may lack sufficient desk space to adequately use a mouse pointer. You may, therefore, want to "go mouseless" by naming the keyboard (or some other input device) as the pointer.

To go mouseless, you need to have the proper configuration specified in the `X*devices` file and to have a special configuration file named `X*pointerkeys`. The default `X*pointerkeys` file is `X0pointerkeys` in the system directory. In light of your experience with `X0screens` and `X0devices`, you will probably recognize this as no mere coincidence.

The `X*pointerkeys` file lets you specify:

- The keys that move the pointer.
- The keys that act as pointer buttons.
- The increments for movement of the pointer.
- The key sequence that resets X11.
- The pixel threshold that must be exceeded before the server switches screens.
- That button chording is enabled or disabled.
- That button latching is enabled or disabled.
- Tablet subsetting.
- Screen switching behavior for multi-screen configurations.

If you modify a `X*pointerkeys` file, it does not take effect until you restart the X Window System again.

### Configuring 'X*devices' for Mouseless Operation

If you have only one keyboard and no pointer devices on the HP-HIL, and you want the keyboard to serve as both keyboard and pointer, you don't have to change the default configuration of `X0devices`. The default input device configuration automatically assigns the pointer to the keyboard if a pointer can't be opened by the server.

If you have two or more input devices, you may need to explicitly specify which device should be the keyboard and which the pointer.

## The Default Values for the 'X*pointerkeys' File

By default, when you configure your keyboard as the pointer, the X server chooses certain number pad keys and assigns them mouse operations. Some number pad keys are assigned to pointer movement; other number pad keys are assigned to button operations.

If you don't need to change the pointer keys from their default specifications, you don't need to do anything else to use your keyboard as both keyboard and pointer. However, if you need to change the default pointer keys, you must edit the `X0pointerkeys` file or create a new `X*pointerkeys` file. The `X*pointerkeys` file is the file that specifies which keys are used to move the pointer when you use the keyboard as the pointer.

The default key assignments are listed in the tables in the following section on customizing the `X*pointerkeys` file.

## Creating a Custom 'X*pointerkeys' File

You need to modify the existing `X0pointerkeys` file only if one or more of the following statements are true:

- You want to use the keyboard for a pointer.
- You want to change the pointerkeys from their default configuration.
- You use the `X0screens` file to configure your display.

You need to create a custom `X*pointerkeys` file only if the following statements are true:

- You want to use the keyboard for a pointer.
- You want to change the pointerkeys from their default configuration.
- You use a configuration file other than the `X0screens` file to configure your display.

### Syntax

You assign a keyboard key to a mouse function (pointer movement or button operation) by inserting a line in the `X*pointerkeys` file. Lines in the `X*pointerkeys` file have the syntax:

*function keyname* [# *comment*]

## Assigning Mouse Functions to Keyboard Keys

You can assign any mouse function, either a pointer movement or a button operation, to any keyboard key. However, make sure that the key you are assigning doesn't already serve a vital function.

You can assign keyboard keys to pointer directions by specifying options in an `X*pointerkeys` file. The following table lists the pointer movement options, the `X*pointerkeys` functions that control them, and their default values:

**Table 9-3. Pointer Movement Functions.**

| Movement Option | Function | Default Key |
|---|---|---|
| Move the pointer to the left. | `pointer_left_key` | keypad_1 |
| Move the pointer to the right. | `pointer_right_key` | keypad_3 |
| Move the pointer up. | `pointer_up_key` | keypad_5 |
| Move the pointer down. | `pointer_down_key` | keypad_2 |
| Add a modifier key to the pointer direction keys. | `pointer_key_mod1` | no default |
| Add a second modifier key to the pointer direction keys. | `pointer_key_mod2` | no default |
| Add a third modifier key to the pointer direction keys. | `pointer_key_mod3` | no default |

Note that the pointer direction keys are the key*pad* number keys on the right side of the keyboard, not the key*board* number keys above the text character keys.

You can assign keyboard keys to pointer distances by specifying options in a X0pointerkeys file. The following table lists the options that determine the distance of pointer movements, the X*pointerkeys functions that control them, and their default value:

**Table 9-4. Pointer Distance Functions.**

| Movement | Function | Default |
|---|---|---|
| Move the pointer a number of pixels. | pointer_move | 10 pixels |
| Move the pointer using a modifier key. | pointer_mod1_amt | 40 pixels |
| Move the pointer using a modifier key. | pointer_mod2_amt | 1 pixel |
| Move the pointer using a modifier key. | pointer_mod3_amt | 5 pixels |
| Add a modifier to the distance keys. | pointer_amt_mod1 | no default |
| Add a modifier to the distance keys. | pointer_amt_mod2 | no default |
| Add a modifier to the distance keys. | pointer_amt_mod3 | no default |

You can assign keyboard keys to mouse button operations by specifying options in a X*pointerkeys file. The following table lists the button operations, the X*pointerkeys functions that control them, and their default values:

**Table 9-5. Button Operation Functions.**

| Button Operation | Function | Default Key |
|---|---|---|
| Perform button 1 operations. | pointer_button1_key | keypad_* |
| Perform button 2 operations. | pointer_button2_key | keypad_/ |
| Perform button 3 operations. | pointer_button3_key | keypad_+ |
| Perform button 4 operations. | pointer_button4_key | keypad_- |
| Perform button 5 operations. | pointer_button5_key | keypad_7 |

You can change the mapping of buttons on the pointer by using options in the X*pointerkeys file. The following table lists the X*pointerkeys functions that control button mapping and their default values. Like xmodmap and xset, these functions affect only the X pointer, not any extension input devices.

**Table 9-6. Button Mapping Functions.**

| Button Mapping | Function | Default Key |
|---|---|---|
| Set button 1 value | button_1_value | 1 |
| Set button 2 value | button_2_value | 2 |
| Set button 3 value | button_3_value | 3 |
| Set button 4 value | button_4_value | 4 |
| Set button 5 value | button_5_value | 5 |

You can change the key sequence that exits the X Window System. Also, if you use both image and overlay planes, you can change the distance you must move the pointer before you switch planes. The following table lists these options, the X*pointerkeys functions that control them, and their default values:

**Table 9-7. Reset and Threshold Functions.**

| Option | Function | Default Key |
|---|---|---|
| Exit the X Window System | `reset` | break |
| Add a modifier to the exit key. | `reset_mod1` | control |
| Add a modifier to the exit key. | `reset_mod2` | left_shift |
| Add a modifier to the exit key. | `reset_mod3` | no default |
| Set the threshold for changing between screens. | `screen_change_amt` | 30 pixels 0 if a graphics tablet is used |

`screen_change_amt` is used only if your system is configured for more than one screen. (Refer to "Using Custom Screen Configurations" in Chapter 3). `screen_change_amt` enables you to avoid switching from one screen to another if you accidentally run the pointer off the edge of the screen. `screen_change_amt` establishes a "distance threshold" that the pointer must exceed before the server switches screens. As the previous table shows, the default width of the threshold is 30 pixels, but acceptable values range from 0 to 255.

When a graphics tablet is used as the X pointer, the `screen_change_amt` defines an area at the left and right edges of the tablet surface that will be used to control screen changes. Moving the puck or stylus into the left or right area will cause the X server to switch to the previous or next screen.

**Table 9-8. Button Chording**

| Option | Function | Default |
|---|---|---|
| Turn button chording off or on. | `button_chording` | ON for devices with 2 buttons, OFF for devices with >2 buttons. |

Button chording refers to the generation of a button by pressing two other buttons. If you have a two-button mouse, you can generate button 3 by pressing both buttons together. With a three-button mouse, you can generate button 4 by pressing the left and middle buttons together and button 5 by pressing the middle and right buttons together. See the button chording examples in the X*pointerkeys file.

You can also use the X*pointerkeys file to configure pointer buttons so they are latched. When this feature is enabled, a button you press stays logically down until you press it again. See the example X*pointerkeys file in your system directory for an example of this functionality.

---

**Note** The sample X*pointerkeys file is placed in /usr/lib/X11 at install time. If you subsequently update your system, the X*pointerkeys file in /usr/lib/X11 is *not* overwritten, and the sample file is placed in /etc/newconfig.

---

**Table 9-9. Specifying a Portion of a Tablet**

| Option | Function | Default |
|---|---|---|
| Use a subset of the tablet surface as the X pointer device | tablet_subset_width | disabled |
| | tablet_subset_height | |
| | tablet_subset_xorigin | |
| | tablet_subset_yorigin | |

If a tablet is used as the X pointer device, it may be desirable to use only a portion of the tablet surface. A rectangular subset of the surface may be specified with these functions. The units are in millimeters from the upper left corner of the tablet surface. For example, if you want to use only an "A" size portion of a larger "B" size tablet, the following lines could be added to the X*pointerkeys file:

```
tablet_subset_xorigin   68
tablet_subset_yorigin   40
tablet_subset_width     296
tablet_subset_height    216
```

You can also use the `X*pointerkeys` file to control screen switching behavior in multi-screen configurations. See the example `X*pointerkeys` file in your system directory for an example of this functionality.

**Note** The sample `X*pointerkeys` file is placed in `/usr/lib/X11` at install time. If you subsequently update your system, the `X*pointerkeys` file in `/usr/lib/X11` is *not* overwritten, and the sample file is placed in `/etc/newconfig`.

## Modifier Keys

You can select up to three keys from among the two (Shift) keys, the two (Extend char) keys, and the (CTRL) key and use them each as **modifier keys.** A modifier key is a key that, when you hold it down and press another key, changes the meaning of that other key.

Modifier keys in the `X*pointerkeys` file have three functions:

- They specify that a certain operation can't take place until they are pressed.
- They enable you to adjust the distance covered by the pointer during a movement operation.
- They enable you to change the key sequence that exits you from X11.

For example, you can overcome the problem in the last example by assigning the (Left Shift) key as a modifier to the pointer direction keys. Now, to move the *hpterm cursor* to the right, you press (▶) as usual. To move the *x server pointer* to the right, you press (Left Shift) (▶).

## Specifying Pointer Keys

The following table lists the valid keynames to use when assigning keyboard keys to mouse functions. You may also use the *default* X Keysymbol names assigned to these keys by the X Server.

**Table 9-10. Valid Pointer Keynames for HP 46021 Keyboards.**

| | | | | | |
|---|---|---|---|---|---|
| **Typewriter Keys:** | | | | | |
| 1 | A | K | U | left_shift | return |
| 2 | B | L | V | left_extend | , |
| 3 | C | M | W | - | . |
| 4 | D | N | X | = | / |
| 5 | E | O | Y | backspace | right_shift |
| 6 | F | P | Z | [ | space_bar |
| 7 | G | Q | ‘ | ] | right_extend |
| 8 | H | R | tab | \ | |
| 9 | I | S | caps_lock | ; | |
| 0 | J | T | control | ’ | |
| **Function Keys:** | | | | | |
| f1 | f3 | f5 | f7 | blank_f9 | blank_f11 |
| f2 | f4 | f6 | f8 | blank_f10 | blank_f12 |
| **Keypad Keys:** | | | | | |
| keypad_1 | keypad_4 | keypad_7 | keypad_0 | keypad_+ | keypad_comma |
| keypad_2 | keypad_5 | keypad_8 | keypad_* | keypad_- | keypad_tab |
| keypad_3 | keypad_6 | keypad_9 | keypad_/ | keypad_enter | keypad_period |
| **Special Keys:** | | | | | |
| enter | stop | clear_line | delete_line | home_cursor | cursor_up |
| escape | menu | clear_display | insert_char | prev | next |
| break | system | insert_line | delete_char | select | cursor_left |
| cursor_down | cursor_right | | | | |

## Examples

If you only have one keyboard and no mouse, and you can live with the default pointer key assignations, you don't have to do anything else to configure your system for mouseless operation. To move the pointer to the left 10 pixels, you would press the 1 key on the keypad. To press mouse button 1 you would press the * key on the keypad.

However, suppose you wanted to move only one pixel to the left. Although the default value of pointer_mod2_amt is one pixel, no key is assigned to the modifier for that amount. Thus, you would need to edit the X0pointerkeys file (or create an X*pointerkeys) to include a line assigning one of the modifier keys to pointer_amt_mod2. The following line in X0pointerkeys assigns the Left Shift key to pointer_amt_mod2:

```
###pointerfunction      key
pointer_amt_mod2     left_shift
```

Or suppose you wanted to set up your X0pointerkeys file so that you could move 1, 10, 25, and 100 pixels. The following lines show one way to specify this:

```
###pointer function       key
pointer_amt_mod1       left_extend
pointer_amt_mod2       left_shift
pointer_amt_mod3       control
pointer_move           1_pixels
pointer_mod1_amt       10_pixels
pointer_mod2_amt       25_pixels
pointer_mod3_amt       100_pixels
```

With these lines in effect, one press of the 1 key on the keypad moves the pointer 1 pixel to the left. Pressing the left Extend char and 1 moves the pointer 10 pixels to the left. Pressing Left Shift 1 moves the pointer 25 pixels to the left. And pressing CTRL 1 moves the pointer 100 pixels to the left.

Or, take the case previously mentioned where you want to use the arrow keys for both text cursor and mouse pointer. You could insert the following lines in your X0pointerkeys file:

```
###pointer function      key
pointer_key_mod1       left_shift
pointer_left_key       cursor_left
pointer_right_key      cursor_right
pointer_up_key         cursor_up
pointer_down_key       cursor_down
```

The above lines enable you to use the arrow keys for cursor movement, while using the shifted arrow keys for pointer movement. Note that only the (Left Shift) key (and not the (Right Shift)) modifies the press of an arrow key from cursor to pointer movement.

Now, suppose you want to use the arrow keys to operate the pointer, and you also need the arrow keys to control the cursor in an hpterm window. Furthermore, another application uses the shift-arrow key sequence to control its cursor.

The easiest way to solve this dilemma is to call in another modifier. The following lines illustrate this. Compare them to the previous example.

```
###pointer function      key
pointer_key_mod1       left_shift
pointer_key_mod2       left_extend
pointer_left_key       cursor_left
pointer_right_key      cursor_right
pointer_up_key         cursor_up
pointer_down_key       cursor_down
```

In this example,

- Pressing the (▲) key moves the hpterm *text cursor* up.
- Pressing (Left Shift) (▲) moves the *cursor* up in the program you frequently operate.
- Pressing (Left Shift) (Left Extend char) (▲) moves the *pointer* up.

Using a similar technique, you can also reassign the (CTRL) (Left Shift) (Reset) sequence that aborts a session. You can specify the press of a single key or a combination of two, three, or four key presses. Just make sure that the key sequence you select isn't something you're going to type by accident.

## Customizing Keyboard Input

Besides remapping the mouse's pointer and buttons to your keyboard, you can remap any key on the keyboard to any other key.

### Modifying Modifier Key Bindings with 'xmodmap'

To change the meaning of a particular key for a particular X11 session, or to initialize the X server with a completely different set of key mappings, use the xmodmap client.

**Note**

There are now two keyboards available for Hewlett-Packard workstations, the 46021 keyboard, and the C1429 keyboard. See appendix B, Using the Keyboards, for more information on using these keyboards and the differences between them.

The syntax for xmodmap is as follows:

$$\texttt{xmodmap} \left[\begin{array}{l} \texttt{-help} \\ \texttt{-grammar} \\ \texttt{-e}\ \textit{expression} \\ \left\{\begin{array}{l} \texttt{-verbose} \\ \texttt{-quiet} \end{array}\right\} \\ \texttt{-n} \\ \texttt{-p} \\ \texttt{-pm} \\ \texttt{-pk} \\ \texttt{-pp} \\ \texttt{-display}\ \textit{host:display} \\ \texttt{-} \end{array}\right] [\textit{filename}]$$

| | |
|---|---|
| `-display` | Specifies the host, display number, and screen to use. |
| `-help` | Displays a brief description of xmodmap options. |
| `-grammar` | Displays a brief description of the syntax for modification expressions. |
| `-verbose` | Prints log information as xmodmap executes. |
| `-quiet` | Turns off verbose logging. This is the default. |

|  |  |
|---|---|
| `-n` | Lists changes to key mappings without actually making those changes. |
| `-e` | Specifies a remapping expression to be executed. |
| `-pm, -p` | Prints the current modifier map to the standard output. This is the default. |
| `-pk` | Prints the current keymap table to the standard output. |
| `-pp` | Print the current pointer map to the standard output. |
| `-` | Specifies that the standard input should be used for the input file. |
| *filename* | Specifies a particular key mapping file to be used. |

## Specifying Key Remapping Expressions

Whether you remap a single key "on the fly" with a command-line entry or install an entire new keyboard map file, you must use valid expressions in your specification, one expression for each remapping.

A valid expression is any one of the following:

**Table 9-11. Valid 'xmodmap' Expressions.**

| To do this ... | Use this expression ... |
|---|---|
| Assign a key symbol to a keycode. | `keycode` *keycode* = *keysym* |
| Replace a key symbol expression with another. | `keysym` *keysym* = *keysym* |
| Clear all keys associated with a modifier key. | `clear` *modifier* |
| Add a key symbol to a modifier. | add *modifier* = *keysym* |
| Remove a key symbol from a modifier. | `remove` *modifier* = *keysym* |

| | |
|---|---|
| `keycode` | Refers to the numerical value that uniquely identifies each key on a keyboard. Values may be in decimal, octal, or hexadecimal. |
| `keysym` | Refers to the character symbol name associated with a keycode, for example, KP_Add. |
| `modifier` | Specifies one of the eight modifier names. |

The following are the modifier names available for use in keyboard customization:

**Table 9-12. Valid Modifier Names.**

| Modifier Names | | | |
|---|---|---|---|
| Shift | Control | Mod2 | Mod4 |
| Lock | Mod1 | Mod3 | Mod5 |

On Hewlett-Packard HP-UX keyboards, the `mod1` modifier is set to the (Extend char) keys (Meta_L and Meta_R). However, any of the modifiers can be associated with any valid key symbol. Additionally, you can associate more than one key symbol with a modifier (such as Lock = Shift_R and Shift_L), and you can associate more than one modifier with a key symbol (for example, Control = Meta_L and Mod1 = Meta_L).

For example, you can press [d] to print a lower case "d", [Shift] [d] to print a capital "D", [Extend char] [d] to print something else, and [Shift] [Extend char] [d] to print still something else.

The xmodmap client gives you the power to change the meaning of any key at any time or to install a whole new key map for your keyboard.

## Examples

Suppose you frequently press the [Caps] key at the most inopportune moments. You could remove the [Caps] lock key from the lock modifier, swap it for the [f1] key, then map the [f1] key to the lock modifier. Do this is by creating a little swapper file that contains the following lines:

```
!This file swaps the [Caps] key with the [F1] key.

remove Lock = Caps_Lock
keysym Caps_Lock = F1
keysym F1 = Caps_Lock
add Lock = Caps_Lock
```

Note the use of the ! in the file to start a comment line. To put your "swapper" file into effect, enter the following on the command line:

```
xmodmap swapper
```

If you use such a swapper file, you should probably have an unswapper file. The following file enables you to swap back to the original keyboard mapping without having to exit X11:

```
!This file unswaps the [F1] key with the [Caps] key.

remove Lock = Caps_Lock
keycode 88 = F1
keycode 55 = Caps_Lock
add Lock = Caps_Lock
```

Note the use of the hexadecimal values to reinitialize the keycodes to the proper key symbols. You put your "unswapper" file into effect by entering the following command line:

```
xmodmap unswapper
```

On a larger scale, you can change your current keyboard to a Dvorak keyboard by creating a file with the appropriate keyboard mappings.

```
xmodmap .keymap
```

## Printing a Key Map

The -pk option prints a list of the key mappings for the current keyboard.

```
xmodmap -pk
```

The list contains the keycode and up to four 2-part columns. The first column contains unmodified key values, the second column contains shifted key values, the third column contains meta (Extend char) key values, and the fourth column contains shifted meta key values. Each column is in two parts: hexadecimal key symbol value, and key symbol name.

# Printing and Screen Dumps

The X Window System includes clients that enable you to do **screen dumps**. A screen dump is an operation that captures an image from your screen and saves it in a bitmap file. You can then redisplay, edit, or send the file to the printer for hardcopy reproduction.

Read this chapter if you need to "take a picture" of something on the screen for future use or if you want to print what is on your screen.

This chapter discusses the following topics:

- Making a screen dump.
- Displaying a screen dump.
- Printing a screen dump.

## Making and Displaying Screen Dumps

X11 windows can be dumped into files by using the `xwd` client. The files can be redisplayed on the screen by using the `xwud` client.

### Making a Screen Dump with 'xwd'

The `xwd` client allows you to take a "picture" of a window that is displayed on the screen and store it in a file. The filed picture can then be printed, edited, or redisplayed. You select the window to be dumped either by clicking the mouse on it or by specifying the window name or id on the command line.

The resulting file is called an `xwd`-format bitmap file or an `xwd` screen dump. All of the figures used in this manual are `xwd` screen dumps.

The syntax for `xwd` is as follows:

```
    ┌ -help                  ┐
    │ ┌ -id id        ┐      │
    │ { -name name    }      │
    │ └ -root         ┘      │
    │ -add value             │
xwd │ -nobdrs                │
    │ ┌ > filename     ┐     │
    │ └ -out filename  ┘     │
    │ -xy                    │
    └ -display display       ┘
```

| | |
|---|---|
| `-help` | Provides a brief description of usage and syntax. |
| `-id` | Specifies the window to be dumped by its *id* rather than using the mouse to select it. |
| `-add` | Adds value to every pixel. |
| `-name` | Specifies the window to be dumped by its *name* rather than using the mouse to select it. |
| `-root` | Specifies that the window to be dumped is the root window. |
| `-nobdrs` | Dumps the window without borders. |
| `-out` | Specifies that the screen dump is to be stored in the file *filename*. |

| | |
|---|---|
| `>` | Specified that the screen dump is to be stored in the file *filename*. |
| `-xy` | Selects 'XY' format of storage instead of the default 'Z' format. |
| `-display` | Specifies the screen that contains the window to be dumped. |

This first example stores a window in a file named `savewindow`, using the pointer to determine which window you want.

1. Display the an hpterm or xterm window.
2. Type:

   ```
   xwd -out savewindow (Return)
   ```

   The pointer changes shape, signifying you can select a window to dump.
3. Move the pointer into the window you want to dump. Press and release any pointer button. After the image is captured, the cursor changes back to its normal shape and the window is stored in the file `savewindow`.

If you know the name of the window you want to dump, you don't need to use the pointer at all. This example dumps the window named "calendar" to a file named `calendar.dump`.

```
xwd -name calendar -out calendar.dump (Return)
```

## Displaying a Stored Screen Dump with 'xwud'

The `xwud` client allows you to display an `xwd`-format file on your monitor. You could have created the file earlier with `xwd` or translated it from another format into `xwd` format.

**Note** The image to be restored has to match the depth of the display on which it is to be restored. For example, an image created and stored using a depth of four cannot be restored on a display with a different depth.

The syntax for xwud is as follows:

```
     ┌-help                          ┐
xwud │-in filename                   │
     │-inverse                       │
     └-display host:display.screen   ┘
```

| | |
|---|---|
| `-help` | Displays a brief description of the options. |
| `-in` | Specifies the file containing the screen dump. |
| `-inverse` | Reverses black and white from the original monochrome dump. |
| `-display` | Specifies the screen on which to display the dump. |

This example displays the xwd-format file `myfile`.

```
xwud -in myfile Return
```

## Printing Screen Dumps

Before you can print the screen dump, you need to ensure that your printer is connected and talking to your computer.

Refer to the system administrator manual(s) for your system if you need to:

- Connect the printer to your computer.
- Create a device file for the printer on your computer.
- Run the print spooler.

### Printing Screen Dumps with 'xpr'

xpr prints a screen dump that has been produced by xwd.

```
     ┌ -scale scale                   ┐
     │ -density dpi                   │
     │ -height inches                 │
     │ -width width                   │
     │ -left inches                   │
     │ -top inches                    │
     │ -header caption                │
     │ -trailer caption               │
     │ { -landscape }                 │
xpr  │ { -portrait  }                 │ filename
     │ -rv                            │
     │ -compact                       │
     │ { -output filename }           │
     │ { -append filename }           │
     │ -noff                          │
     │ -split n                       │
     │ -device dev                    │
     │ -cutoff level                  │
     └ -noposition                    ┘
```

| | |
|---|---|
| `-scale` | Specifies a multiplier for pixel expansion. The default is the largest that will allow the entire image to fit on the page. |
| `-density` | Specifies the dots per inch for the printer. |

| | |
|---|---|
| `-height` | Specifies the maximum height in inches of the window on the page. |
| `-width` | Specifies the maximum width in inches of the window on the page. |
| `-left` | Specifies the left margin in inches. The default is centered. |
| `-top` | Specifies the top margin in inches. The default is centered. |
| `-header` | Specifies a caption to print above the window. |
| `-trailer` | Specifies a caption to print below the window. |
| `-landscape` | Prints the window in landscape mode. The default prints the long side of the window on the long side of the paper. |
| `-portrait` | Prints the window in portrait mode. The default prints the long side of the window on the long side of the paper. |
| `-rv` | Reverses black and white from the original screen. |
| `-compact` | Provides efficient printer directions for a window with lots of white space (PostScript printers only). |
| `-output` | Specifies a file to store the output in. |
| `-append` | Adds the window to the end of an existing file. |
| `-noff` | Specifies that the window should appear on the same page as the previous window. Used with -append. |
| `-split` | Prints the window on *n* pages. Not applicable to HP printers. |
| `-device` | Specifies the printer to use. |

| | |
|---|---|
| `ljet` | HP LaserJet series, HP ThinkJet, HP QuietJet, RuggedWriter, HP2560 series, HP2930 series, other PCL devices. |
| `pjet` | HP PaintJet (color mode). |
| `pjetxl` | HP PaintJet XL. |
| `ln03` | DEC LN03. |
| `la100` | DEC LA100. |
| `ps` | PostScript printers. |
| `pp` | IBM PP3812. |

| | |
|---|---|
| `-cutoff` | Specifies intensity for converting color to monochrome for printing on a HP LaserJet printer. |

`-noposition` Bypasses header positioning, trailer positioning, and image positioning commands for the HP LaserJet and HP PaintJet printers.

*filename* Specifies the `xwd` file to print.

For example, suppose you want to print a `xwd` file named `myfile` that you previously created with `xwd`. You want to print the file on a HP LaserJet printer in portrait mode with black and white the reverse of the original `xwd` file.

```
xpr -device ljet -portrait -rv myfile | lp -oraw Return
```

Reversing colors is often used when preparing illustrations for documents. The original illustration can be done in white with a black background, which is easy to see on computer displays, but reversed to give a black drawing on a white background, which is common in printed material.

## Moving and Resizing the Image on the Paper

You may not always want to have the image print exactly in the same size or location as the default choices place it.

### Sizing Options

The three sizing options for `xpr` are:

`-scale` Each bit of the image is translated into a grid of the size you specify. For example, if you specify a scale of 5, each bit in the image is translated into a 5 by 5 grid. This is an easy way to increase the size without refiguring the height and width.

`-height` The maximum height in inches of the image on the page.

`-width` The maximum width in inches of the image on the page.

The actual printed size could be smaller than `-height` and `-width` if other options, such as the orientation ones, conflict with them.

### Location Options

The two location options for `xpr` are:

`-left` The left margin in inches.

`-top` The top margin in inches.

If `-left` is not specified, the image is centered left-to-right. If `-top` is not specified, the image is centered top-to-bottom.

### Orientation Options

The two orientation options to xpr are:

| | |
|---|---|
| `-landscape` | The image is printed so that the top of the image is on the long side of the paper. |
| `-portrait` | The image is printed so that the top of the image is on the short side of the paper. |

If neither option is specified, xpr will position the image so that the long side of the image is on the long side of the paper. However, you can force it to print either in landscape mode or portrait mode by using the appropriate option.

Unless told otherwise by the sizing options, xpr makes the image as big as necessary to fit in the orientation specified.

## Printing Multiple Images on One Page

xpr normally prints each image on a separate page. The `-noff` option is used to print more than one image on a page.

## Printing Color Images

Use the device name `pjet` to direct output to a HP PaintJet printer.

For example, the following command prints a `xwd` file named `myfile` on a HP PaintJet printer.

```
   xpr -device pjet myfile (Return)
```

Color images printed on a HP LaserJet printer will be in black and white instead of color.

xpr prints only in black and white, no shades of gray. If your original color image contained many colors of the same intensity, the HP LaserJet printer version may be all light or all dark. If that happens, use the `-cutoff` option to change the mapping of color intensities. Anything above the cutoff value is white and anything below is black. Note that the default cutoff value is 50 percent.

If you want color images to print in shades of gray on your LaserJet, use the Starbase utility `pcltrans` instead of xpr. Refer to the StarBase documentation for information.

11

# Using Starbase

Starbase is a powerful graphics library from Hewlett-Packard. It provides two-dimensional and three-dimensional graphics, a variety of input and output capabilities, and high performance features, such as hidden surface removal, shading, and light sourcing.

This chapter describes how the X Window System interacts with Starbase. It does not describe Starbase itself. For detailed information about using Starbase with X, see *Starbase Graphics Techniques* in the Starbase documentation.

This chapter covers the following topics:

- Setting environment variables for Starbase.
- Using the `X*screens` file to control display options.
- Starting the X server.
- Opening and destroying windows for Starbase applications.
- Creating transparent windows.
- Conversion utilities.

## Using the X*screens File

This section reviews some concepts that you need to understand before starting Starbase applications:

- `X*screens` file.
- Operating modes.
- Double buffering.

The `X*screens` file is a system file that contains the screen configurations you want to use. Before you run a Starbase application, you should ensure that the configuration is correct for Starbase. The `X*screens` file is described in chapter 3.

The following sections describe options you should be aware of when running Starbase with the X Window System. It also explains how to specify those options in the `X*screens` file.

## Operating Modes

A table later in this section describes which options and modes are possible on specific hardware.

### Image and Overlay Planes

Display hardware can have two kinds of display planes, image and overlay. The image plane allows the hardware to help the graphics commands run faster and more efficiently.

### Server Operating Modes

The operating mode results from the way you specify the image and overlay screens in the `X*screens` file.

The four different modes are:

| | |
|---|---|
| Overlay mode | The server operates only in the overlay planes. Starbase can display in its "raw" mode, writing directly to the image planes, rather than to a window. A "transparent" overlay window can look through to the Starbase display in the image planes. Starbase can use the double buffering feature only in the image planes, not the overlay planes. |
| Image mode | This is the only mode available on those displays that do not have overlay planes. Even if overlay planes are available, you may want to use image mode to have a greater number of colors available. |
| Stacked screen mode | In this mode, the image planes are treated as one screen and the overlay planes as another, separate screen, providing twice as much screen space. The pointer is moved to the edge of the display to switch between the overlay and image planes. |
| Combined mode | This mode, when available, treats the overlay and image planes as a single device that provides multiple window types to client programs. |

Monochrome and LOW_COLOR displays run in the image mode.

Documentation for the Starbase application program will tell you which mode or which plane the application expects.

## Double Buffering

Double buffering means that Starbase uses half of the color planes of your displays to display to the screen, and uses the other half to compute and draw the next screen display. This provides smooth motion for animation, and it is also faster. However, double buffering reduces the number of colors available for displaying on the screen at one time. Some applications require double buffering. If you run a double-buffered application in singlebuffer mode, the display will flash or flicker rapidly.

This feature does not apply to monochrome displays or when the X server is running in the overlay planes.

The following example shows an `X*screens` file with appropriate entries for image, stacked, and combined screen modes, while enabling double buffer mode in the image planes:

```
### Image Mode ###
/dev/image_device doublebuffer

### Stacked Screens Mode ###
# /dev/overlay_device
# /dev/image_device doublebuffer

### Combined Screens Mode ###
# /dev/overlay_device /dev/image_device doublebuffer
```

## Screen Depth

You can specify a screen depth for image planes in the `X*screens` file. Valid depths for regular (single buffer) mode are 8 and 24. Valid depths for doublebuffered mode are 8, 16, and 24. The depth of overlay planes is determined by the `/dev` entry in `X*screens`.

More planes means more colors can be displayed simultaneously. For computer-generated graphics to look as realistic as photographs, thousands of colors must be shown at the same time. 8 planes means that $2^8$ (256) colors can be shown, while 24 planes means that $2^{24}$ (16 million) colors can be shown. Note that depth is specified only when you have more than one depth available.

The following example shows an `X*screens` file entry for a display running in image mode. Windows can have 8 planes (256 colors) displayed simultaneously.

```
/dev/image_device depth 8
```

The next following example provides two doublebuffered depths in the image planes: depth 8 (16 planes/2) and depth 12 (24 planes/2). That is, some windows in the image planes could have a depth of 8 planes, while others could have a depth of 12 planes. This is possible only in combined mode.

```
/dev/overlay_device /dev/image_device depth 16 depth 24 doublebuffer
```

## Display Hardware Options

Table 11-1. Display Hardware and Available Options

| **With this display hardware ...** | **You can use these options ...** | | | |
|---|---|---|---|---|
| **HP Part Number** | **Maximum Planes (colors)** | **Modes** | **Double buffer** | **Depth** |
| HP A1096 | 1 Image (monochrome) | Image | | |
| HP A1416 | 2 Overlay (4)<br>8 Image (256) | Image<br>Overlay<br>Stacked | √ | |
| HP A1439 | 8-24 Image (256) | Image | | |
| HP A1659 | 8 Image (256) | Image | | |
| HP A1924 | 8 Image (256) | Image | | |
| HP 9000S300<br>Model 425e | 8 Image (256) | Image | √ | |
| HP 9000S700<br>Model 710 | 8 Image (256) | Image | | |
| HP 98542A | 1 Image (monochrome) | Image | | |
| HP 98543A | 4 Image (16) | Image | √ | |
| HP 98544B | 1 Image (monochrome) | Image | | |
| HP 98545A | 4 Image (16) | Image | | |
| HP 98547A | 6 Image (64) | Image | √ | |
| HP 98548A | 1 Image (monochrome) | Image | | |
| HP 98549A | 6 Image (64) | Image | √ | |
| HP 98550A | 2 Overlay (4)<br>8 Image (256) | Image<br>Overlay<br>Stacked | √ | |

Table 11-1.
Display Hardware and Available Options (continued)

| With this display hardware ... | You can use these options ... | | | |
|---|---|---|---|---|
| **HP Part Number** | **Maximum Planes (colors)** | **Modes** | **Double buffer** | **Depth** |
| HP 98704A | 3/4 Overlay<br>8/16 Image | Image<br>Overlay<br>Combined | <br><br>✓ | <br><br>✓ |
| HP 98720A | 3 Overlay (8)<br>8-24 Image (256) | Image<br>Overlay<br>Stacked | ✓<br><br>✓ | ✓<br><br>✓ |
| HP 98730A | 3/4 Overlay<br>8-24 Image (256) | Image<br>Overlay<br>Stacked<br>Combined | ✓<br><br>✓<br>✓ | ✓<br><br>✓<br>✓ |
| HP 98735A | 3/4 Overlay<br>8-24 Image | Image<br>Overlay<br>Combined | <br><br>✓ | <br><br>✓ |
| HP 98765A | 3/4 Overlay<br>8-24 Image | Image<br>Overlay<br>Combined | <br><br>✓ | <br><br>✓ |

## Starting the Server

Once you have ensured that the options you need are in the `X*screens` file, you must restart the server for the options to take effect.

## Window-Smart and Window-Naive Programs

Window-smart applications are able to create and destroy the windows in which they operate.

Window-naive (sometimes called window-dumb) applications aren't able to create and destroy windows on their own. They need help from the X Window System.

### Is My Application Window-Smart or Window-Naive?

If you are using an existing application, the documentation that comes with the application will tell you how to start it. You don't have to worry whether it is window-smart or window-naive, just follow the directions.

If you are writing a new application using Starbase, use the `xwcreate` and `xwdestroy` commands. Rather than typing the commands each time you want to test the new program, put the commands in a file, then execute the file to start the application. In this case, the application is window-naive but the file is window-smart.

### Running Window-Smart Programs

From an `hpterm` window, type the name of the Starbase program you want to run.

For example, the following command will start a hypothetical Starbase application named `planetarium` that displays a moving view of the night sky. Assume that the program is in the `/users/funstuff` directory on your computer.

```
/users/funstuff/planetarium (Return)
```

### Running Window-Naive Programs

Window-naive programs cannot open and close the window they need to run in, so you must do it for them with clients (a terminal emulator, for example). Some old programs that use the Starbase graphics library are window-naive.

Most window-naive programs are able to run in the X Window System environment using the `sox11` device driver. The `sox11` driver is described in the *Starbase Device Drivers* manual. But window-naive clients still need help to create and destroy the windows they display their output in.

To enable window-naive graphics programs to run within X, you need four special helper clients to create and destroy the windows used by the naive graphics programs. The clients are:

- `gwind` (HP-UX only)
- `xwcreate`
- `xwdestroy`
- `gwindstop` (HP-UX only)

On HP-UX operating systems, `gwind` runs in the background and services requests from the other three helper clients. When requested by `xwcreate`, `gwind` creates a window in which an application can display its output; when requested by `xwdestroy`, `gwind` destroys the window. *You don't need to start the* `gwind` *program,* `xwcreate` *and* `xwdestroy` *start and stop it for you.*

The next sections cover:

- Creating a window
- Destroying a window

## Creating a Window with 'xwcreate'

`xwcreate` requests `gwind` to create a window for a window-naive graphics program to use for its output. The graphics program must exist on the same computer that is running `xwcreate`. If `gwind` is not already running when `xwcreate` is executed, `xwcreate` will start `gwind`. Once `xwcreate` has created a window, you can use the window to run your graphics program. When you finish that application, you can use the same window to run another graphics program if you wish.

Use `xwcreate` from the command line.

```
         ┌ -display host:display.screen        ┐
         │ -parent parent                      │
         │ -geometry width×height±col±row      │
         │ -r                                  │
         │ -bg color                           │
         │ -bw pixels                          │
xwcreate │ -bd color                           │
         │ -depth depth                        │
         │ -visual visualclass                 │
         │ -overlay                            │
         │ -wmdir directory                    │
         └ -title name                         ┘
```

| | |
|---|---|
| `-display` | Specifies the screen the window will appear on |
| `-parent` | Names a window to be the parent of the window being created. |
| `-geometry` | Specifies desired size and location of window. |
| `-r` | Specifies backing store. Default is no backing store. |
| `-bg` | Specifies the background color. The default is black. |
| `-bw` | Specifies the border width in pixels. The default is 3 pixels wide. |
| `-bd` | Specifies the border color. The default is white. |
| `-depth` | Specifies the depth of the window. The default is the same depth as its parent. |
| `-visual` | Specifies the visual class of the window when multiple visual classes are supported by the display at the specified depth. |
| `-overlay` | Specifies that an overlay plane visual should be used. |
| `-wmdir` | Specifies the name of the directory containing the pty file for the window. |
| `-title` | Specifies the name the window will be called. |

The `depth` option is where you tell the window manager what set of planes you want the window to be in. If you specify nothing, the window is created with the same depth as its parent, or with the same depth as the root if no parent is specified. If you specify a depth, the window will be placed in the image plane with the depth (number of color planes) you specify.

The following example creates a window named "foo:"

```
xwcreate -title foo (Return)
```

## Destroying a Window with 'xwdestroy'

`xwdestroy` destroys the window created by `xwcreate`. If that window is the only graphics window present at that time, `gwind` will also terminate.

Use `xwdestroy` from the command line.

`xwdestroy` [`-wmdir` *path/directory*] *window1 window2* ...

| | |
|---|---|
| `-wmdir` | Specifies the directory containing the pty file for the window. |
| *window* | Specifies the window or windows to be destroyed. |

The following example will destroy a window named "foo:"

```
xwdestroy foo
```

### Destroying a Window with 'gwindstop'

`gwindstop` is used on HP-UX systems only.

`gwindstop` destroys all windows created by `gwind` in the specified directory. If, however, you use `xwdestroy` to remove the *last* window opened for graphics use, `xwdestroy` will terminate `gwind`. You *do not* need to use `gwindstop`.

**Caution** You must use `xwdestroy` or `gwindstop` to get rid of a window after you have finished running your graphics application. Do *not* use `kill` to remove the `gwind` process associated with the window. If you should accidentally do so, you must type the command `rm $WMDIR/wm`. Failure to do this will result in `xwcreate` not running the next time you call it.

Use `gwindstop` from the command line.

```
gwindstop [directory] [directory] ...
```

| | |
|---|---|
| *directory* | The directory containing the pty files for the windows to be destroyed. |

## Running Starbase in Raw Mode

If your display supports overlay planes, you can run Starbase in "raw" mode, which means that Starbase writes to the entire screen rather than to a window. You then use a transparent window to see through to the Starbase output.

For information about Starbase raw mode, refer to the Starbase documentation.

## Using Transparent Windows

Transparent windows allow you to look through an overlay window into the image planes.

### Creating a Transparent Window with 'xseethru'

`xseethru` is a transparent overlay-plane window used to see through the overlay planes to the image planes.

Use `xseethru` from the command line.

$$\texttt{xseethru} \begin{bmatrix} \texttt{-geometry}\ \textit{width}\times\textit{height}\pm\textit{col}\pm\textit{row} \\ \texttt{-display}\ \textit{host:display.screen} \end{bmatrix}$$

| | |
|---|---|
| `-geometry` | The geometry used to create the window. |
| `-display` | The screen the window will appear on. |

This example opens a transparent window 100-pixels by 100-pixels in size and located 50 pixels from the left and 25 pixels from the top of the screen.

```
xseethru -geometry 100x100+50+25 (Return)
```

### Creating a Transparent Window with 'xsetroot'

`xsetroot` allows you to make the root window transparent when you are running X in the overlay planes.

Use `xsetroot` from the command line.

`xsetroot` [`-solid` *color*]

| | |
|---|---|
| `-solid` | Sets the window color to *color*. |

This example turns the root window into a transparent window.

```
xsetroot -solid transparent (Return)
```

### Creating a Transparent Background Color

Any window may have `transparent` as its background color.

This example opens an `hpterm` window with a transparent background color.

```
hpterm -bg transparent (Return)
```

## Conversion Utilities

This section shows you how to use the utilities `sb2xwd` and `xwd2sb`.

### Converting Starbase Format to 'xwd' Format using 'sb2xwd'

`sb2xwd` converts Starbase format window files into `xwd` format pixmaps. The pixmaps can then be printed by using `xpr` or displayed on the screen by using `xwud`, both of which are described in chapter 8.

Use `sb2xwd` from the command line.

The syntax for `sb2xwd` is:

`sb2xwd` < *filename* > *filename*

| | |
|---|---|
| <*filename* | The Starbase window file to be converted. |
| >*filename* | The `xwd` pixmap file name. |

This example translates the Starbase window file named `mystar` into an `xwd` pixmap file named `myxwd`, then prints it on an HP LaserJet printer.

```
sb2xwd < mystar > myxwd
xpr -dev ljet myxwd | lp -oraw
```

### Converting 'xwd' Format to Starbase Format using 'xwd2sb'

`xwd2sb` is the opposite of `sb2xwd`. It converts `xwd` format pixmaps into Starbase format window files.

Use `xwd2sb` from the command line.

The syntax for `xwd2sb` is:

`xwd2sb` < *filename* >*filename*

| | |
|---|---|
| <*filename* | The `xwd` bitmap file to be converted. |
| >*filename* | The Starbase window file filename. |

This example dumps a window named `sample` into a `xwd` file called `myxwd`, translates it into a Starbase window file called `mystar`, and prints it using the Starbase `pcltrans` utility.

```
xwd -name sample -out myxwd
xwd2sb < myxwd > mystar
pcltrans mystar | lp -oraw
```

A

# Reference Information

This section contains reference information (man pages) for the X server, clients, and utility programs included with the X Window System. The entries are arranged alphabetically, each starting on its own "page 1."

Hewlett-Packard has selected a set of the X Consortium clients to support for the HP-UX release 9.0. Some clients that were supported in earlier releases are no longer supported. This is due to current and announced future level of X Consortium support for certain clients, as well as new functionality in R5. In addition, several new clients are not supported

Six clients supported in 8.* releases are not supported for the 9.0 release. Five of these clients are supplied in the 9.0 release, but may not be supplied in the future. These clients are:

| | |
|---|---|
| `bitmap` | Replaced by `vueicon`, the VUE icon editor. If you use the unsupported 9.0 version of `bitmap`, note that it is substantially different from the 8.* version. See the `bitmap` man page for complete information. |
| `xclipboard` | This function was a sample implementation. You can use a terminal emulator and an editor to get the same functionality. |
| `xcutsel` | The functionality of `xcutsel` is now contained in `hpterm`. |
| `xfd` | You can display font graphics on any application by specifying `-fn` as a command-line option. You can display font metrics by using `xlsfonts`. |
| `xrefresh` | The functionality of `xrefresh` is now handled by the window manager. |

The sixth client, `bdftosnf`, is not supplied with 9.0 since it has been replaced by `bdftopcf`.

The new clients not supported are `columns`, `dr_vue`, `xdpyinfo`, `xprop`, and `vuefincalc`.

The executable files and man pages for all the unsupported clients are installed in `/usr/contrib/bin/X11` and `/usr/contrib/man`, respectively. In order to use these clients, be sure that your $PATH environment variable contains `/usr/contrib/bin/X11`. Use `env` to see the current setting of your environment variables. The online man page access utility `man` looks in `/usr/contrib/man` by default, so you don't need to modify the $MANPATH environment variable.

**Table A-1. Man Pages for Supported Clients**

| | |
|---|---|
| bdftopcf(1) | xhost(1) |
| fs(1) | xinit(1) |
| gwind(1) | xinitcolormap(1) |
| gwindstop(1) | xload(1) |
| hpterm | xlsfonts(1) |
| mkfontdir(1) | xmodmap(1) |
| mwm(1) | xpr(1) |
| resize(1) | xrdb(1) |
| rgb(1) | xseethru(1) |
| sb2xwd(1) | X(1) |
| stconv(1) | Xserver(1) |
| stlicense(1) | xset(1) |
| stload(1) | xsetroot(1) |
| stmkdirs(1) | xterm(1) |
| stmkfont(1) | xwcreate(1) |
| vueicon(1) | xwd(1) |
| vuelogin(1X) | xwd2sb(1) |
| x11start(1) | xwdestroy(1) |
| xauth(1) | xwininfo(1) |
| xclock(1) | xwud(1) |
| xcmsdb | |

**Table A-2. Man Pages for Unsupported Clients**

| | |
|---|---|
| bitmap(1) | xcutsel(1) |
| columns(6) | xdpyinfo(1) |
| dr_vue(1) | xfd(1) |
| vuefincalc(1) | xprop(1) |
| xclipboard(1) | xrefresh(1) |

## NAME

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

## SYNOPSIS

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

## DESCRIPTION

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

## OPTIONS

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

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

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

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

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

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

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

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

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

## SEE ALSO

X(1)

## COPYRIGHT

Copyright 1991, Massachusetts Institute of Technology.
See *X(1)* for a full statement of rights and permissions.

## AUTHOR

Keith Packard, MIT X Consortium

**NAME**

fs - X font server

**SYNOPSIS**

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

**DESCRIPTION**

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

**STARTING THE SERVER**

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

A

**OPTIONS**

**-config configuration_file**
: Specifies the configuration file the font server will use.

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

**-port tcp_port**
: Specifies the TCP port number on which the server will listen for connections.

**-daemon**
: Runs the font server as a daemon.

**SIGNALS**

*SIGTERM*
: This causes the font server to exit cleanly.

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

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

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

**CONFIGURATION**

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

Recognized keywords include:

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

: The current Font Server implementation supports only a single catalogue ("all"), containing all of the specified fonts. Future implementations will support multiple catalogues that can be selectively accessed by different Font Server clients.

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

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

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

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

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

error-file (string)
: Filename of the error file to be used if syslog(3) logging (below) is disabled. If error-file is not specified and syslog logging is disabled, all messages are sent to stderr.

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

use-syslog (boolean)
: If true (the default), causes errors to be logged via the syslog(3) facility.

## EXAMPLE

```
#
# sample font server configuration file
#

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

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

# alternate font servers for clients to use
alternate-servers = host1:7001,host1:7002

# where to look for fonts
# the first is a set of Type1 outlines, the second is a set of
# misc bitmaps and the last is another font server
#
catalogue = /usr/lib/fonts/type1.st,
        /usr/lib/X11/fonts/misc,
        tcp/host2:7000

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

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

## FONT SERVER NAMES

One of the following forms can be used to name a font server that accepts TCP connections:

tcp/[*hostname*]:*port*
tcp/[*hostname*]:*port/cataloguelist*

The *hostname* specifies the name (or decimal numeric address) of the machine on which the font server is running -- if not specified, it defaults to the current machine. The *port* is the decimal TCP port on which the font server is listening for connections. The *cataloguelist* specifies a list of catalogue names, with '+' as a separator.

Examples: *tcp/expo.lcs.mit.edu:7000, tcp/18.30.0.212:7001/all, tcp/:7001.*

One of the following forms can be used to name a font server that accepts DECnet connections:

decnet/*nodename*::font$*objname*
decnet/*nodename*::font$*objname/cataloguelist*

The *nodename* specifies the name (or decimal numeric address) of the machine on which the font server is running. The *objname* is a normal, case-insensitive DECnet object name. The *cataloguelist* specifies a list of catalogue names, with '+' as a separator.

Examples: *DECnet/SRVNOD::FONT$DEFAULT*, *decnet/44.70::font$special/symbols*.

**SEE ALSO**

X(1), *Font server implementation overview*

**BUGS**

Multiple catalogues should be supported.

**COPYRIGHT**

A

**AUTHORS**

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

## NAME

gwind - graphics window daemon

## SYNOPSIS

**gwind**

## DESCRIPTION

The *gwind* program creates X windows under the command of the utility programs *xwcreate, xwdestroy,* and *gwindstop.* The window can then be accessed by Starbase application program (see *xwcreate(1m)*). The daemon remains in operation until all windows created via *xwcreate* are destroyed, or until the *gwindstop* utility is executed.

A

## FILES

| | |
|---|---|
| **/dev/screen** | directory created to contain the pty devices used to communicate between the utility programs and the gwind daemon. |

## DEPENDENCIES

Implemented on the Series 300 and 800 only.

## SEE ALSO

gwindstop(1m), xwcreate(1m), xwdestroy(1m)

## AUTHOR

HP

## NAME

gwindstop - terminate the window helper facility

## SYNOPSIS

**gwindstop [directory] [directory] ..**

## DESCRIPTION

**gwindstop**

destroys windows and their associated pty files from named directories. The windows must have been created earlier by xwcreate(1).

A

**directory**

is the name of the directory where the pty files for the windows reside. If **directory** name is not supplied, */dev/screen* is taken to be the desired directory. Otherwise, if the directory argument implies an absolute pathname, then it will be taken to be the desired directory. Otherwise, the directory name will be taken to be relative to the value of the environment variable $WMDIR. If $WMDIR is not defined in the environment, the directory name will be taken to be relative to */dev/screen.* Note: if $WMDIR is defined in the environment, it must represent an absolute pathname.

## DIAGNOSTICS

If the windows in the indicated directory are successfully destroyed, then the program remains silent. If one or more directories could not be found, an error message ( "Invalid directory") will be printed on the standard output.

## ORIGIN

HP

## SEE ALSO

xwcreate(1), xwdestroy(1).

## NAME

hpterm - X window system Hewlett-Packard terminal emulator.

## SYNOPSIS

**hpterm** [-toolkitoption] [-option]

## DESCRIPTION

The *hpterm* program is a terminal emulator for the X Window system. It provides a Term0 compatible terminal for programs that can't use the window system directly. It also emulates many of the block mode features of HP terminals. Refer to the WARNINGS section below for additional information about running block mode applications.

## OPTIONS

The *hpterm* terminal emulator accepts all of the standard X Toolkit command line options along with additional options all of which are listed below (if the option begins with a '+' instead of a '-', the option is restored to its default value):

**-b** *number*
This option specifies the size of the inner border (the distance between the outer edge of the character and the window border) in pixels. Associated resource: ***borderWidth.**

**-background** *color*
This option specifies the color to use for the background of the window. Associated resource: ***background.**

**-bd** *color*
This option specifies the color to use for the border of the window. Associated resource: ***borderColor.**

**-bg** *color*
This option specifies the color to use for the background of the window. Associated resource: ***background.**

**-borderwidth** *number*
This option specifies the width in pixels of the border surrounding the window. Associated resource: ***TopLevelShell.borderWidth.**

**-bs** This option indicates that the "background" of the term0 text entry window should be the select color that corresponds to the specified background color. Associated resource: ***backgroundIsSelect.**

**+bs** This option indicates that the "background" of the term0 text entry window should be the specified background. Associated resource: ***backgroundIsSelect.**

**-bw** *number*
This option specifies the width in pixels of the border surrounding the window. Associated resource: ***TopLevelShell.borderWidth.**

**-cr** *color* This option specifies the color to use for the text cursor. Associated resource: ***cursorColor.**

**-display** *display*
This option specifies the X server to contact; see *X*(1). Associated resource: none.

**-e** *command* [*arguments* ...]
This option specifies the command (and its command line arguments) to be run in the *hpterm* window. The default is to start the user's shell. **This must be the last option on the command line.** Associated resource: none.

**-fb** *font* This option specifies a font to be used when displaying bold (alternate) text. This font must be the same height and width as the normal (primary) font. If only one of the normal (primary) or bold (alternate) fonts is specified, it will be used for both fonts. Refer to the NLS section. Associated resource: ***boldFont.**

**-fg** *color* This option specifies the color to use for displaying text. Associated resource: ***foreground.**

**-fn** *font* This option specifies a font to be used when displaying normal (primary) text. If only one of the normal (primary) or bold (alternate) fonts is specified, it will be used for both fonts. Refer to the NLS section. Associated resource: ***font.**

**-font** *font*
This option specifies a font to be used when displaying normal (primary) text. If only one of the normal (primary) or bold (alternate) fonts is specified, it will be used for both fonts. Associated resource: ***font.**

A

**-foreground** *color*
This option specifies the color to use for displaying text. Associated resource: ***foreground.**

**-geometry** *geometry*
This option specifies the preferred size and position of the *hpterm* window; see *X*(1). Associated resource: ***term0.geometry.**

**-help** This option will display a help message. Associated resource: none.

**-i** This option indicates that *hpterm* should supply the window manager with a bitmapped icon. Associated resource: **bitmapIcon.**

**+i** This option indicates that the window manager should generate its own icon for *hpterm.* Associated resource: **bitmapIcon.**

**-iconic** This option indicates that *hpterm* should be placed on the display in icon form. Associated resource: ***term0.iconic.**

**+iconic** This option indicates that *hpterm* should not be placed on the display in icon form. Associated resource: ***term0.iconic.**

**-kshmode**
This option indicates that *hpterm* should convert characters entered with the extend key pressed into a two character sequence consisting of an ASCII escape followed by the unextended character. Associated resource: ***kshMode.**

**-l** This option indicates that *hpterm* should send all terminal output to a log file as well as to the screen. Logging may not be enabled when the -L option is used. Associated resource: ***logging.**

**+l** This option indicates that *hpterm* should not do logging. Associated resource: ***logging.**

**-lf** *file* This option specifies the name of the file to which the output log described above is written. If *file* begins with a pipe symbol (|), the rest of the string is assumed to be a command to be used as the endpoint of a pipe. The default filename is **HptermLog***XXXXX* (where *XXXXX* is the process id of *hpterm*) and is created in the directory from which *hpterm* was started. Associated resource: ***logFile.**

**-ls** This option indicates that the shell that is started in the *hpterm* window should be a login shell (i.e. the first character of argv[0] will be a dash, indicating to the shell that it should read the user's /etc/profile and .profile (for ksh and sh) or /etc/csh.login and .login (for csh). Associated resource: ***loginShell.**

**+ls** This option indicates that the shell that is started should not be a login shell (i.e. it will be a normal "subshell"). Associated resource: ***loginShell.**

**-map** This option indicates that *hpterm* should map (de-iconify) itself upon pty output if it is unmapped (iconified). An initial period of time during which *hpterm* will not map itself upon pty output may be specified via the **mapOnOutputDelay** resource. Associated resource: ***mapOnOutput.**

**+map** This option indicates that *hpterm* should not map (de-iconify) itself upon pty output if it is unmapped (iconified). Associated resource: ***mapOnOutput.**

**-mb** This option indicates that the pointer cursor should be put into blanking mode. In this mode, the cursor will turn on when the pointer is moved, and will be blanked either after a selectable number of seconds or after keyboard input has occurred. The delay is set via the **pointerBlankDelay** resource. Associated resource: ***pointerBlank.**

**+mb** This option indicates that the pointer cursor should remain on. Associated resource: ***pointerBlank.**

**-mc** *mode*
This option determines how *hpterm* will generate the foreground color, shadow colors, and shadow tiles of the scrollbar and softkey widgets. Valid modes are "all", "shadow", and "none." Associated resource: ***makeColors.**

**-ms** *color*
This option specifies the color to be used for the pointer cursor. Associated resource: ***pointerColor.**

A

**-name** *name*
This option specifies the application name under which resources are to be obtained, rather than the default executable file name ("hpterm"). Associated resource: **.name.**

**-reverse** This option indicates that reverse video should be simulated by swapping the foreground and background colors. Associated resource: ***reverseVideo.**

**-rv** This option indicates that reverse video should be simulated by swapping the foreground and background colors. Associated resource: ***reverseVideo.**

**+rv** This option indicates that reverse video should not be simulated. Associated resource: ***reverseVideo.**

**-sb** This option indicates that a scrollbar should be displayed. Associated resource: ***scrollBar.**

**+sb** This option indicates that a scrollbar should not be displayed. Associated resource: ***scrollBar.**

**-sbbg** *color*
This option specifies the color to use for the background of the scrollbar window. Associated resource: ***scrollBar.background.**

**-sbfg** *color*
This option specifies the color to use for the foreground of the scrollbar window. This value will be ignored if the **makeColors** resource is set to **all.** Associated resource: ***scrollBar.foreground.**

**-skbg** *color*
This option specifies the color to use for the background of the softkey window. Associated resource: ***softkey.background.**

**-skfg** *color*
This option specifies the color to use for displaying softkey text. This value will be ignored if the **makeColors** resource is set to **all.** Associated resource: ***softkey.foreground.**

**-skfn** *font*
This option specifies a font to be used when displaying softkey text. Associated resource: ***softkey.font.**

**-sl** *number*[*suffix*]
This option indicates the number of off screen lines to be saved in the terminal buffer. If no suffix is included or the suffix is l, the total length of the terminal buffer will be *number* plus the length of the terminal window. If the suffix is s the total length of the terminal buffer will be (*number* plus one) times the length of the terminal window. Associated resource: ***saveLines.**

**-ti** *name* This option specifies a name for *hpterm* to use when identifying itself to application programs. Refer to the WARNINGS section for additional information about using *hpterm* with block mode applications. Associated resource: ***termId.**

**-title** *name*
: This option specifies a window title for *hpterm*. This string may be used by the window manager when displaying the application. Associated resource: **.TopLevelShell.title.**

**-tm** *string*
: This option specifies a string containing terminal-setting keywords and the characters to which they may be bound. Associated resource: ***ttyModes.**

**-tn** *name*
: This option specifies a name for *hpterm* to set the **$TERM** environment variable to. Associated resource: ***termName.**

A

**-vb** This option indicates that a visual bell is preferred over an audible one. Instead of ringing the terminal bell whenever a Control-G is received, the window will be flashed. Associated resource: ***visualBell.**

**+vb** This option indicates that a visual bell should not be used. Associated resource: ***visualBell.**

**-xrm** *resourcestring*
: This option specifies a resource string to be used. This is especially useful for setting resources that do not have separate command line options. Associated resource: none.

**-C** This option indicates that the window should receive console output. The server must be authorized to receive console output. See "XConsoles" below for additional information. Associated resource: none.

-*Sccn* This option specifies the last two letters of the name of a pseudoterminal to use in slave mode, and the file descriptor of the pseudoterminal's master. This allows *hpterm* to be used as an input and output channel for an existing program and is sometimes used in specialized applications such as *pam*(1). This option will only work with pty names of the form "ttyxx." For example, "-S p01" specifies "ttyp0" on file descriptor 1. Associated resource: none.

-*Spty.fd* This option specifies the unique portion of the name of a pseudoterminal to use in slave mode, and the file descriptor of the pseudoterminal's master. This allows *hpterm* to be used as an input and output channel for an existing program and is sometimes used in specialized applications such as *pam*(1). This option will work for all pty names. For example, "-S p0.1" specifies "ttyp0" on file descriptor 1 and "-S p02.13" specifies "ttyp02" on file descriptor 13. Associated resource: none.

**-U** Reserved for internal use.

**-W** Reserved for internal use.

The following command line arguments are provided for compatibility with older versions. They may not be supported in future releases as the X Toolkit provides standard options that accomplish the same task.

*#geometry*
: This option specifies the preferred position of the icon window. It is shorthand for specifying the ***iconGeometry** resource. Associated resource: **.iconGeometry.**

-T *string* This option specifies the title for *hpterm*'s window. It is equivalent to **-title** *string*. Associated resource: **.TopLevelShell.title.**

**-n** *string* This option specifies the icon name for *hpterm*'s windows. It is shorthand for specifying the ***iconName** resource. Associated resource: ***iconName.**

**-r** This option indicates that reverse video should be simulated by swapping the foreground and background colors. It is equivalent to **-reversevideo** or **-rv**. Associated resource: ***reverseVideo.**

**+r** This option indicates that reverse video should not be simulated. It is equivalent to **+rv**. Associated resource: ***reverseVideo.**

**-w** *number*
: This option specifies the width in pixels of the border surrounding the window. It is

equivalent to **-borderwidth** *number* or **-bw** *number.* Associated resource: ***TopLevelShell.borderWidth.**

## RESOURCES

The *hpterm* window consists of a Motif shell widget which contains a form widget. The form widget contains a term0 widget, scrollbar widget, and softkey widget. Resources specific to the shell widget are:

**hpterm Resource Set**

| Name | Class | Type | Default |
|---|---|---|---|
| borderColor | BorderColor | Pixel | black |
| borderWidth | BorderWidth | int | 2 |
| geometry | Geometry | string | |
| iconGeometry | IconGeometry | string | |
| name | Name | string | hpterm |
| title | Title | string | Terminal emulator |

A

**borderColor**
This resource defines the border color of the *hpterm* window.

**borderWidth**
This resource specifies the width of the *hpterm* window border. This value may be modified by the window manager.

**geometry**
This resource specifies the preferred size and position of the *hpterm* window.

**iconGeometry**
This resource specifies the preferred size and position of *hpterm* when iconified. It is not necessarily obeyed by all window managers.

**name** This resource specifies the name of the instance of the program. It is used when extracting resources from the resource database.

**title** This resource specifies the window title for *hpterm*. This string may be used by the window manager when displaying this application.

A

**term0 Resource Set**

| Name | Class | Type | Default |
|---|---|---|---|
| allowSendEvents | AllowSendEvents | Boolean | FALSE |
| background | Background | Pixel | "white" |
| backgroundIsSelect | BackgroundIsSelect | string | FALSE |
| bitmap | Bitmap | string | |
| bitmapIcon | BitmapIcon | Boolean | FALSE |
| boldFont | Font | string | see NLS below |
| copyLine | CopyLine | string | "right" |
| cursorColor | Foreground | Pixel | "black" |
| cut | Cut | string | "left" |
| dc1Handshake | Dc1Handshake | DC1Handshake | "auto" |
| flashBorder | FlashBorder | Boolean | FALSE |
| font | Font | string | see NLS below |
| foreground | Foreground | Pixel | "black" |
| *fn*Attribute | SoftkeyAttribute | int | 2 |
| *fn*Label | SoftkeyLabel | string | see below |
| *fn*String | SoftkeyString | string | see below |
| halfBrightInhibit | HalfBrightInhibit | Boolean | FALSE |
| iconic | Iconic | Boolean | FALSE |
| internalBorder | BorderWidth | int | 2 |
| keyboardLanguage | KeyboardLanguage | string | see NLS below |
| keyboardLanguageList | KeyboardLanguageList | string | see NLS below |
| kshMode | KshMode | Boolean | FALSE |
| logFile | LogFile | string | "HptermLog*XXXXX*" |
| logging | Logging | Boolean | FALSE |
| loginShell | LoginShell | Boolean | FALSE |
| makeColors | MakeColors | string | "none" |
| mapOnOutput | AutoMap | Boolean | FALSE |
| mapOnOutputDelay | MapDelay | int | 0 |
| paste | Paste | string | "middle" |
| pointerBlank | PointerBlank | Boolean | FALSE |
| pointerBlankDelay | PointerBlankDelay | int | 3 |
| pointerColor | Foreground | Pixel | "black" |
| pointerShape | PointerShape | string | "xterm" |
| reverseVideo | ReverseVideo | Boolean | FALSE |
| roman8 | Roman8 | Boolean | TRUE |
| saveLines | SaveLines | string | "1s" |
| scrollBar | ScrollBar | Boolean | FALSE |
| softkeyInitialize16 | SoftkeyInitialize16 | Boolean | FALSE |
| softkeySelect | SoftkeySelect | string | "left" |
| stickyNextCursor | StickyCursor | Boolean | TRUE |
| stickyPrevCursor | StickyCursor | Boolean | TRUE |
| termId | TermId | string | "X-hpterm" |
| termName | TermName | string | "hpterm" |
| ttyModes | TtyModes | string | none |
| visualBell | VisualBell | Boolean | FALSE |

**allowSendEvents**
This resource defines whether synthetic key and button events (generated using the X protocol SendEvent request) should be interpreted or discarded.

**background**
This resource defines the background color of the text window.

**backgroundIsSelect**
This resource controls the color used as the "background" of the term0 text entry window and defaults to False. When False, the background is the color specified. When True, the

background is the "select color" that corresponds to the background. For visual consistency with other Motif-based applications, set this resource to True.

**bitmap** This resource defines whether *hpterm* will override its built in bitmap icon with a user specified bitmap icon. If the path does not begin with a "/" or "./", it will be processed relative to "/usr/lib/X11/bitmaps".

**bitmapIcon**
This resource defines whether *hpterm* will supply the window manager with a bitmapped icon. The supplied bitmap may be ignored by the window manager.

**boldFont**
This resource defines the font used for bold (alternate) text. See "NLS" below for defaults.

**copyLine**
This resource defines the pointer button/modifier combination to be used to activate the CopyLine function. See "POINTER USAGE" below.

**cursorColor**
This resource defines the text cursor color. The pointer cursor color is defined by the **pointerColor** resource.

**cut** This resource defines the pointer button/modifier combination to be used to activate the Cut function. See "POINTER USAGE" below.

**dc1Handshake**
This resource controls how DC1/DC2 handshaking is enabled for transferring data between hpterm and the application requesting the data. This resource can have one of three values: "auto", "enabled", or "disabled". When set to "disabled", no handshaking is required for transferring data from hpterm to the application; when set to "enabled", the handshaking mode is controlled strictly by the state of the InhHndShk (strap G) and InhDC2 (strap H) flags. Setting **dc1Handshake** to "auto" causes hpterm to start out with DC1 handshaking initially disabled, it is automatically enabled the first time hpterm receives a DC1 character and is once again controlled by the state of the InhHndShk (strap G) and InhDC2 (strap H) flags.

**flashBorder**
This resource defines whether *hpterm* window border will change color when the pointer cursor enters or leaves the window.

**font** This resource defines the font used for normal (primary) text. See the "term0.*fontLanguage* (class Term0.FontLanguage) Resource Set" table and "NLS" below for defaults.

**foreground**
This resource defines the foreground (text) color of the text window.

***fn*Attribute**
This resource defines the softkey attribute for softkey *n* (1 - 16). If "softkeyInitialize16" is true, all 16 softkey attributes can be initialized. If it false, only the first 8 softkey attributes can be initialized.

***fn*Label**
This resource defines the softkey label for softkey *n* (1 - 16). If "softkeyInitialize16" is true, all 16 softkey labels can be initialized. If it false, only the first 8 softkey labels can be initialized. The default labels for softkeys 1 - 8 are "f1" - "f8." The default labels for softkeys 9 - 16 are empty.

***fn*String**
This resource defines the softkey string for softkey *n* (1 - 16). If "softkeyInitialize16" is true, all 16 softkey strings can be initialized. If it false, only the first 8 softkey strings can be initialized. The default strings for softkeys 1 - 8 are "<esc>p" - "<esc>w." The default labels for softkeys 9 - 16 are empty.

**halfBrightInhibit**
This resource defines whether half-bright enhancements will be not be generated. When true, full-bright characters will be used instead of half-bright characters.

**iconic** This resource defines whether *hpterm* will start up in iconic form.

**internalBorder**
This resource defines the number of pixels between the characters and the window border.

A

**keyboardLanguage**
This resource defines the default keyboard language *hpterm* should use. See "NLS" below for details and defaults.

**keyboardLanguageList**
This resource defines the list of keyboard languages that may be selected from the terminal configuration menu. See "NLS" below for detail and defaults.

**kshMode**
This resource defines whether *hpterm* will operate in ksh mode. In ksh mode, *hpterm* converts characters entered with the extend key pressed into a two-character sequence consisting of an ASCII escape followed by the un-extended character.

**logFile** This resource defines the name of the file to which a terminal session is logged. The default is "**HptermLog***XXXXX*" (where *XXXXX* is the process id of *hpterm*).

**logging** This resource defines whether a terminal session will be logged. It is also available at runtime via the Device Control menu. Logging may not be enabled when the "-L" option is used.

**loginShell**
This resource defines whether the shell to be run in the window will be started as a login shell (i.e., the first character of argv[0] will be a dash, indicating to the shell that it should read the user's /etc/profile and .profile (for ksh and sh) or /etc/csh.login and .login (for csh).

**makeColors**
This resource is provided for backward compatibility with older versions of hpterm; since it may not be supported in future releases, it is no longer recommended for use. This resource defines how the **bottomShadowColor**, **foreground**, and **topShadowColor** resources of the scrollbar and softkey widgets will be generated and how the **foreground** resource of the term0 widget will be generated. If the value of this resource is "all", then *hpterm* will use the value of the **background** resource of the term0 widget to generate a value for the **foreground**, and the **background** resource of the softkey and scrollbar widgets to generate values for the **bottomShadowColor**, **foreground**, and **topShadowColor** resources such that there is a 3-D look. In this case the **topShadowTile** and **bottomShadowTile** are always set to "foreground." If the **makeColors** resource value is "shadow" the **bottomShadowColor** and **topShadowColor** will be generated but **foreground** will not be generated. If the **makeColors** resource value is set to "none" then no colors will be generated.

**mapOnOutput**
This resource defines whether *hpterm* will map (de-iconify) itself upon pty output if it is unmapped (iconified). An initial period of time during which *hpterm* will not map itself upon pty output may be specified to allow *hpterm* to not map itself upon initial shell output. The delay is set via the **mapOnOutputDelay** resource.

**mapOnOutputDelay**
This resource defines the number of seconds at startup during which *hpterm* will not map (de-iconify) itself upon pty output.

**paste** This resource defines the pointer button/modifier combination to be used to activate the Paste function. See "POINTER USAGE" below.

**pointerBlank**
This resource defines whether *hpterm* will put the pointer cursor into blanking mode. In blanking mode, the pointer cursor will turn on when the pointer is moved, and will be blanked either after a selectable number of seconds or after keyboard input has occurred. The delay is set via the **pointerBlankDelay** resource.

**pointerBlankDelay**
This resource defines the number of seconds to wait before blanking the pointer cursor after the pointer has been moved. When set to "0", the pointer will be blanked only upon keyboard input.

**pointerColor**
This resource defines the pointer cursor color. The text cursor color is defined by the **cursorColor** resource.

**pointerShape**
This resource defines the pointer cursor shape. Valid cursor shapes may be found in the file "/usr/include/X11/cursorfont.h." Shapes are specified as the name with the leading "XC_" dropped. Valid cursor shapes include "left_ptr", "crosshair", and "xterm."

**reverseVideo**
This resource defines whether reverse video will be simulated by swapping the foreground and background colors.

**roman8**
This resource controls the mapping of keys to characters and is effective only for western european keyboards. Roman8 encoding is used when set to TRUE, ISO 8859-1 encoding is used when set to FALSE. (It is the user's responsibility to ensure that correctly encoded fonts are used; refer to the discussion on fonts in your **Using the X Window System Manual** for more information on font characteristics.)

**saveLines**
This resource defines the number of lines in the terminal buffer beyond the length of the window. The resource value consists of a "*number*" followed by an optional "*suffix*." If no *suffix* is included or the *suffix* is "l" the total length of the terminal buffer will be *number* plus the length of the terminal window. If the suffix is "s" the total length of the terminal buffer will be (*number* plus one) times the length of the terminal window. *Hpterm* **will try to maintain the same buffer to window ratio when the window is resized larger.**

**scrollBar**
This resource defines whether the scrollbar will be displayed.

**softkeyInitialize16**
This resource enables initialization of all 16 softkeys. If false, only the first 8 softkeys can be initialized via resources.

**softkeySelect**
This resource defines the pointer button/modifier combination to be used for selecting softkeys. See "POINTER USAGE" below.

**stickyNextCursor**
This resource defines whether the cursor should be homed when the **Next** key is pressed. When true, the cursor will be in the same screen position after the key is pressed that it was in before pressing the key. When false, the cursor will be moved to the upper left hand corner of the screen after the key is pressed.

**stickyPrevCursor**
This resource defines whether the cursor should be homed when the **Prev** key is pressed. When true, the cursor will be in the same screen position after the key is pressed that it was in before pressing the key. When false, the cursor will be moved to the upper left hand corner of the screen after the key is pressed.

**termId** This resource defines the name for *hpterm* to use when identifying itself to application programs. Refer to the WARNINGS section for additional information about using *hpterm* with block mode applications.

**termName**
This resource defines the string for set the "$TERM" environment variable.

**ttyModes**
This resource specifies a string containing terminal-setting keywords and the characters to which they may be bound. Allowable keywords include: intr, quit, erase, kill, eof, eol, swtch, start, stop, brk, susp, dsusp, rprnt, flush, weras, and lnext. Control characters may be specified as ^char (e.g. ^c or ^u), and ^? may be used to indicate delete. This is very useful for overriding the default terminal settings without having to do an *stty* every time an *hpterm* is started.

A

**visualBell**
This resource defines whether a visible bell (i.e. flashing) should be used instead of an audible bell when Control-G is received.

**term0.*fontLanguage* (class Term0.FontLanguage) Resource Set**

| Name | Class | Type | Default |
|---|---|---|---|
| primary.high | FontPosition.Size | string | see NLS below |
| primary.medium | FontPosition.Size | string | see NLS below |
| primary.low | FontPosition.Size | string | see NLS below |
| alternate.high | FontPosition.Size | string | see NLS below |
| alternate.medium | FontPosition.Size | string | see NLS below |
| alternate.low | FontPosition.Size | string | see NLS below |

*fontLanguage*.**primary.high**
This resource defines the default normal (primary) font for displays with high resolution monitors. See "NLS" below for additional information.

*fontLanguage*.**primary.medium**
This resource defines the default normal (primary) font for displays with medium resolution monitors. See "NLS" below for additional information.

*fontLanguage*.**primary.low**
This resource defines the default normal (primary) font for displays with low resolution monitors. See "NLS" below for additional information.

*fontLanguage*.**alternate.high**
This resource defines the default bold (alternate) font for displays with high resolution monitors. See "NLS" below for additional information.

*fontLanguage*.**alternate.medium**
This resource defines the default bold (alternate) font for displays with medium resolution monitors. See "NLS" below for additional information.

*fontLanguage*.**alternate.low**
This resource defines the default bold (alternate) font for displays with low resolution monitors. See "NLS" below for additional information.

The following resources are specified as part of the "softkey" widget (name "softkey", class "Softkey"). For example, the softkey font resource would be specified one of:

```
HPterm*softkey*font:   hp8.8x16
HPterm*Softkey*font:   hp8.8x16
*Softkey*Font:         hp8.8x16
```

Additional resources and information can be found in the **XmPrimitive**(3X) and **CORE**(3X) man pages along with additional information about the various shadow options.

**Softkey Resource Set**

| **Name** | **Class** | **Type** | **Default** |
|---|---|---|---|
| background | Background | Pixel | "white" |
| bottomShadowColor | Foreground | Pixel | "black" (see below) |
| bottomShadowTile | BottomShadowTile | string | "foreground" (see below) |
| font | Font | string | (see below) |
| foreground | Foreground | Pixel | "black" (see below) |
| topShadowColor | Background | Pixel | "white" (see below) |
| topShadowTile | TopShadowTile | string | "50_foreground" (see below) |

**background**
This resource defines the background color of the softkey window.

**bottomShadowColor**
This resource defines the color that is combined with the bottom shadow tile and foreground color to create a pixmap used to draw the bottom and right sides of the softkey borders. This may be overridden by the term0 **makeColors** resource described above.

**bottomShadowTile**
This resource defines the tile used in creating the pixmap used for drawing the bottom and right shadows for the softkey borders. Valid tile names are described in **XmCreateTile**(3X). This may be overridden by the term0 **makeColors** resource described above.

**font** This resource defines the font used for softkey text. The softkey font will default to the normal (primary) font of the text window.

**foreground**
This resource defines the foreground (text) color of the softkey window. This may be overridden by the term0 **makeColors** resource described above.

**topShadowColor**
This resource defines the color that is combined with the top shadow tile and foreground color to create a pixmap used to draw the top and left sides of the softkey borders. This may be overridden by the term0 **makeColors** resource described above.

**topShadowTile**
This resource defines the tile used in creating the pixmap used for drawing the top and left shadows for the softkey borders. Valid tile names are described in **XmCreateTile**(3X). This may be overridden by the term0 **makeColors** resource described above.

The following resources are specified as part of the "Xmscrollbar" widget (name "scrollBar", class "ScrollBar"). Some example scrollbar resources are:

```
HPterm*scrollBar*initialDelay:   10
HPterm*ScrollBar*RepeatRate:     10
*ScrollBar*Granularity:          1
hpterm*scrollBar*width:          20
```

Additional resources and information can be found in the **XmPrimitive**(3X), **XmScrollBar**(3X), **XmValuator**(3X), and **Core**(3X) man pages along with additional information about the various shadow options.

**Scrollbar Resource Set (name "scrollBar", class "ScrollBar")**

| Name | Class | Type | Default |
|---|---|---|---|
| background | Background | Pixel | "white" |
| bottomShadowColor | Foreground | Pixel | "black" (see below) |
| bottomShadowTile | BottomShadowTile | string | "foreground" (see below) |
| foreground | Foreground | Pixel | "black" (see below) |
| granularity | Granularity | int | 2 |
| initialDelay | InitialDelay | int | 500 |
| repeatRate | RepeatRate | int | 100 |
| topShadowColor | Background | Pixel | "white" (see below) |
| topShadowTile | TopShadowTile | string | "50_foreground" (see below) |
| width | Width | int | 10 |

A

**background**
This resource defines the background color of the scrollbar window.

**bottomShadowColor**
This resource defines the color that is combined with the bottom shadow tile and foreground color to create a pixmap used to draw the bottom and right sides of the scrollbar borders. This may be overridden by the term0 **makeColors** resource described above.

**bottomShadowTile**
This resource defines the tile used in creating the pixmap used for drawing the bottom and right shadows for the scrollbar borders. Valid tile names are described in **XmCreateTile**(3X). This may be overridden by the term0 **makeColors** resource described above.

**foreground**
This resource defines the foreground color of the scrollbar window. This may be overridden by the term0 **makeColors** resource described above.

**granularity**
This resource defines the number of lines to advance the slider when the button is being held down on an arrow. The value is defined in milliseconds.

**initialDelay**
This resource defines the delay to wait between the time the button is held down on an arrow before the slider starts its repetitive movement. The value is defined in milliseconds.

**repeatRate**
This resource defines the continuous repeat rate to use to move the slider while the button is being held down on an arrow. The value is also defined in milliseconds.

**topShadowColor**
This resource defines the color that is combined with the top shadow tile and foreground color to create a pixmap used to draw the top and left sides of the scrollbar borders. This may be overridden by the term0 **makeColors** resource described above.

**topShadowTile**
This resource defines the tile used in creating the pixmap used for drawing the top and left shadows for the scrollbar borders. Valid tile names are described in **XmCreateTile**(3X). This may be overridden by the term0 **makeColors** resource described above.

**width** This resource defines the width of the scrollbar in pixels.

## POINTER USAGE

*Hpterm* allows you to cut and paste text within its own or other windows. All cutting and pasting is done using the PRIMARY selection. (To maintain compatability with previous versions of hpterm (and other applications that use cut buffers), the cutting and pasting is also done to/from the first global cut buffer. When pasting, *hpterm* gets its text from the PRIMARY selection; if the PRIMARY selection is not owned, or the current owner cannot supply the data as text, *hpterm* will try to get its data from the first global cut buffer.) The PRIMARY selection will be disowned (and the selected text unhighlighted) under the following conditions:

1) the cursor is moved anywhere before the end of the selected region
2) the beginning of the selected region is scrolled off the end of the terminal buffer
3) the selected region is scrolled across the boundaries of the locked region when memory lock is enabled.

The default button assignments may be changed via various resource strings. The cut and paste functions and their default button assignments are:

**Cut** The left button is used to "cut" text into the cut buffer. Move the pointer to the beginning of the text to cut, press the button, move the cursor to the end of the region, and release the button. The "cut" text will not include the character currently under the pointer.

**Paste** The middle button "pastes" the text from the cut buffer, inserting it as keyboard input.

**CopyLine** The right hand button "cuts" the text from the pointer (at button release) through the end of line (including the new line), saving it in the cut buffer, and immediately "pastes" the line, inserting it as keyboard input. This provides a history mechanism.

The copyLine, cut, and paste key functions can be configured to any button and modifier combination desired via various resources. Each assignment consists of an optional combination of modifiers ("none" or any combination of "shift", "meta", "lock", "control", "mod1", ..., "mod5" separated by blanks), followed by a "|" and the name of the button ("left", "middle", "right", "button1", ..., "button5"). For example, if it is desired for the cut function to be associated with the middle button with shift and control pressed, one could use the following resource line:

```
*cut:                    shift control | middle
```

For a full list of resource names, see "RESOURCES" above.

## NLS

*Hpterm* currently supports 23 different language versions of the HP keyboard. It is possible to switch between different languages via the "terminal configuration" menu. A list of language to choose from along with their order is specified via the "keyboardLanguageList" resource. The "keyboardLanguageList" resource consists of a list of keyboard languages separated by spaces, tabs, or new lines. Valid keyboard languages may be found in the file "/usr/lib/X11/XHPlib.h." Keyboard languages are specified as the language with the leading "KB_" dropped. The default value for the "keyboardLanguageList" resource is "US_English Belgian Canada_English Danish Dutch Finnish French Canada_French Swiss_French German Swiss_German Italian Norwegian Euro_Spanish Latin_Spanish Swedish UK_English Katakana Swiss_French2 Swiss_German2 Japanese Korean S_Chinese T_Chinese."

The initial keyboard language is specified via the "keyboardLanguage" resource. If the string is NULL, the language of the server's keyboard will be used. If the keyboard language specified is not included in the keyboardLanguageList resource, the first language included in the keyboardLanguageList resource will be used. The default is to use the language of the server's keyboard.

*Hpterm* will try to select default fonts which match your monitor and your keyboard language. If the normal (primary) and bold (alternate) fonts are specified, they will be used. If only one is specified (via either command line options or resources), it will be used for both the normal (primary) and bold (alternate) fonts. If neither normal (primary) or bold (alternate) fonts are specified, *hpterm* trys to find them based on the default keyboard language. The default keyboard language is indicated on the "terminal configuration" menu. The built in defaults may be overridden via the "term0.*fontLanguage*" resources. *FontLanguage* varies with the keyboard language as follows.

| **keyboard language** | *fontLanguage* |
|---|---|
| Katakana | hp_kana8 |
| Japanese | hp_japanese |
| Korean | hp_korean |
| T_Chinese | hp_chinese_t |
| S_Chinese | hp_chinese_s |
| all others HP keyboards | hp_roman8 |
| non HP keyboards | iso_8859_1 |

A

The default font size used will depend upon the resolution of the monitor as follows:

| **monitor resolution** | **font size** |
|---|---|
| 72 DPI or less | low |
| greater than 72 DPI and less 100 DPI | medium |
| 100 DPI or greater | high |

For example, resource specifications for the US_English, German, and Finnish keyboards would be:

```
HPterm*hp_roman8.primary.high:       *courier-medium-r-normal--14*hp-roman8
HPterm*hp_roman8.primary.medium:     *courier-medium-r-normal--12*hp-roman8
HPterm*hp_roman8.primary.low:        *courier-medium-r-normal--8*hp-roman8
HPterm*hp_roman8.alternate.high:     *courier-bold-r-normal--14*hp-roman8
HPterm*hp_roman8.alternate.medium:   *courier-bold-r-normal--12*hp-roman8
HPterm*hp_roman8.alternate.low:      *courier-bold-r-normal--8*hp-roman8
```

For the Japanese keyboard, resource specifications would be:

```
HPterm*hp_japanese.primary.high:       jpn.8x18
HPterm*hp_japanese.primary.medium:     jpn.8x18
HPterm*hp_japanese.primary.low:        jpn.8x18
HPterm*hp_japanese.alternate.high:     math.8x18
HPterm*hp_japanese.alternate.medium:   math.8x18
HPterm*hp_japanese.alternate.low:      math.8x18
```

If these fonts can not be found, the font "fixed" will be used for both the normal (primary) and bold (alternate) fonts. These resources are for font defaults only and will be ignored if either the normal (primary) or bold (alternate) fonts are specified.

Control-N will switch to the bold (alternate) font and control-O will switch back to the normal (primary) font. *Hpterm* will switch back to the normal (primary) font automatically at the beginning of each line.

## XCONSOLES

It is possible to configure a system to allow redirection of console output (and input) to an *hpterm* window. If the "-C" option is used, *hpterm* will redirect console output (and input) if:

*Hpterm* is displaying on the local system. *Hpterm* must be running on the same system as the server that is displaying the window.

The display is authorized. The display number of the display name (see "Display Specification" in *X*(1)) must be authorized to take control of the console via the file "/usr/lib/X11/Xconsoles." The file is parsed as follows:

A '#' and all following text on a line are ignored.

Blank lines are ignored.

Leading tabs and spaces are ignored.

A number matching the display number authorizes the server to redirect console output (and input).

An asterisk ('*') matches all display numbers and authorizes the server to redirect console output (and input).

If either condition is not met, a warning will be written to stderr and console output (and input) will not be redirected.

## WARNINGS

When running block mode applications, it may be necessary for *hpterm* to identify itself to application programs as some terminal other than "X-hpterm." Most applications understand the terminal id "2392A." Newer applications also understand the terminal id "700/92" while older applications may only understand the terminal id "2622A." To set the terminal identification string, use the "-ti" command line option, the "termId" resource, or the "TermId" class.

The overflow protect mode of memory lock is not supported.

## ENVIRONMENT

*Hpterm* sets the environment variables "$LINES" and "$COLUMNS" to the number of lines and columns of the terminal screen. It also uses and sets the environment variable "$DISPLAY" to specify its server connection. The *resize*(1) command may be used to reset "$LINES" and "$COLUMNS" after the window size has been changed.

## ORIGIN

Hewlett-Packard Company ITO.

## SEE ALSO

*X(1), resize(1), xset(1), xterm(1), pty(4), Core(3X), XmScrollBar(3X), XmPrimitive(3X), XmForm(3X)*

## NAME

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

## SYNOPSIS

**mkfontdir** [directory-names]

## DESCRIPTION

For each directory argument, *mkfontdir* reads all of the bitmapped font files in the directory, searching for properties named "FONT", or (failing that) the name of the file stripped of its suffix. These are used as font names, which are written out to the file **fonts.dir** in the directory along with the name of the font file. The **fonts.dir** file is then used by the X server and the Font server to determine what fonts are available.

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

## SCALABLE FONTS

*Mkfontdir* does not automatically recognize font files containing scalable fonts. The *stmkdirs(1)* utility can be used in place of *mkfontdir* to generate **fonts.dir** files that include entries for Intellifont and Type 1 scalable fonts in addition to entries for bitmapped fonts.

*Mkfontdir* offers another mechanism for "recognizing" scalable fonts: a **fonts.scale** file can be constructed (the format is identical to that in **fonts.dir**) containing entries for scalable fonts. Whenever *mkfontdir* is run on a directory, it copies entries from **fonts.scale** in that directory into the **fonts.dir** it constructs in that directory. This is a useful mechanism for creating *fonts.dir* entries for types of scalable fonts that are not recognized by *stmkdirs*.

## FONT NAME ALIASES

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

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

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

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

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

## USAGE

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

## SEE ALSO

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

**NAME**

mwm – The Motif Window Manager

**SYNOPSIS**

mwm [ options ]

**DESCRIPTION**

mwm is an X Window System client that provides window management functionality and some session management functionality. It provides functions that facilitate control (by the user and the programmer) of elements of window state such as placement, size, icon/normal display, and input-focus ownership. It also provides session management functions such as stopping a client.

**Options**

This option specifies the display to use; see X(1). This option specifies a resource string to use. This option causes mwm to manage all screens on the display. The default is to manage only a single screen. This option causes mwm to retrieve its resources using the specified name, as in name*resource. This option specifies the resource names to use for the screens managed by mwm. If mwm is managing a single screen, only the first name in the list is used. If mwm is managing multiple screens, the names are assigned to the screens in order, starting with screen 0. Screen 0 gets the first name, screen 1 the second name, and so on.

A

**Appearance**

The following sections describe the basic default behaviors of windows, icons, the icon box, input focus, and window stacking. The appearance and behavior of the window manager can be altered by changing the configuration of specific resources. Resources are defined under the heading "X DEFAULTS."

**Screens**

By default, mwm manages only the single screen specified by the -display option or the DISPLAY environment variable (by default, screen 0). If the -multiscreen option is specified or if the multiScreen resource is True, mwm tries to manage all the screens on the display.

When mwm is managing multiple screens, the -screens option can be used to give each screen a unique resource name. The names are separated by blanks, for example, -screens mwm0 mwm1. If there are more screens than names, resources for the remaining screens will be retrieved using the first name. By default, the screen number is used for the screen name.

**Windows**

Default mwm window frames have distinct components with associated functions: In addition to displaying the client's title, the title area is used to move the window. To move the window, place the pointer over the title area, press button 1 and drag the window to a new location. By default, a wire frame is moved during the drag to indicate the new location. When the button is released, the window is moved to the new location. The title bar includes the title area, the minimize button, the maximize button, and the window menu button. In shaped windows, such as round windows, the title bar floats above the window. To turn the window into an icon, click button 1 on the minimize button (the frame box with a small square in it). To make the window fill the screen (or enlarge to the largest size allowed by the configuration files), click button 1 on the maximize button (the frame box with a large square in it). The window menu button is the frame box with a horizontal bar in it. To pull down the window menu, press button 1. While pressing, drag the pointer on the menu to your selection, then release the button when your selection is highlighted. Pressing button 3 in the title bar or resize border handles also posts the window menu. Alternately, you can click button 1 to pull down the menu and keep it posted; then position the pointer and select. You can also post the window menu by pressing <Shift><Esc> or <Alt><Space>. Double-clicking button 1 with the pointer on the window menu button closes the window. The following table lists the contents of the window menu.

| **Default Window Menu** | | |
|---|---|---|
| **Selection** | **Accelerator** | **Description** |
| Restore | Alt+F5 | Restores the window to its size before minimizing or maximizing |
| Move | Alt+F7 | Allows the window to be moved with keys or mouse |
| Size | Alt+F8 | Allows the window to be resized |
| Minimize | Alt+F9 | Turns the window into an icon |
| Maximize | Alt+F10 | Makes the window fill the screen |
| Lower | Alt+F3 | Moves window to bottom of window stack |
| Close | Alt+F4 | Causes client to terminate |

A

To change the size of a window, move the pointer over a resize border handle (the cursor changes), press button 1, and drag the window to a new size. When the button is released, the window is resized. While dragging is being done, a rubber-band outline is displayed to indicate the new window size. An optional matte decoration can be added between the client area and the window frame. A matte is not actually part of the window frame. There is no functionality associated with a matte.

**Icons**

Icons are small graphic representations of windows. A window can be minimized (iconified) using the minimize button on the window frame. Icons provide a way to reduce clutter on the screen.

Pressing mouse button 1 when the pointer is over an icon causes the icon's window menu to pop up. Releasing the button (press + release without moving mouse = click) causes the menu to stay posted. The menu contains the following selections:

| **Icon Window Menu** | | |
|---|---|---|
| **Selection** | **Accelerator** | **Description** |
| Restore | Alt+F5 | Opens the associated window |
| Move | Alt+F7 | Allows the icon to be moved with keys |
| Size | Alt+F8 | Inactive (not an option for icons) |
| Minimize | Alt+F9 | Inactive (not an option for icons) |
| Maximize | Alt+F10 | Opens the associated window and makes it fill the screen |
| Lower | Alt+F3 | Moves icon to bottom of icon stack |
| Close | Alt+F4 | Removes client from mwm management |

Note that pressing button 3 over an icon also causes the icon's window menu to pop up. To make a menu selection, drag the pointer over the menu and release button 3 when the desired item is highlighted.

Double-clicking button 1 on an icon invokes the f.restore_and_raise function and restores the icon's associated window to its previous state. For example, if a maximized window is iconified, then double-clicking button 1 restores it to its maximized state. Double-clicking button 1 on the icon box's icon opens the icon box and allows access to the contained icons. (In general, double-clicking a mouse button is a quick way to perform a function.) Pressing <Shift><Esc> or <Menu> (the pop-up menu key) causes the icon window menu of the currently selected icon to pop up.

**Icon Box**

When icons begin to clutter the screen, they can be packed into an icon box. (To use an icon box, mwm must be started with the icon box configuration already set.) The icon box is a mwm window that holds client icons. It includes one or more scroll bars when there are more window icons than the icon box can show at the same time.

Icons in the icon box can be manipulated with the mouse. The following table summarizes the behavior of this interface. Button actions apply whenever the pointer is on any part of the icon. Note that double-clicking an icon in the icon box invokes the f.restore_and_raise function.

| Button Action | Description |
|---|---|
| Button 1 click | Selects the icon |
| Button 1 double-click | Normalizes (opens) the associated window<br>Raises an already open window to the top of the stack |
| Button 1 drag | Moves the icon |
| Button 3 press | Causes the menu for that icon to pop up |
| Button 3 drag | Highlights items as the pointer moves across the menu |

Pressing mouse button 3 when the pointer is over an icon causes the menu for that icon to pop up.

| Icon Menu for the Icon Box | | |
|---|---|---|
| **Selection** | **Accelerator** | **Description** |
| Restore | Alt+F5 | Opens the associated window (if not already open) |
| Move | Alt+F7 | Allows the icon to be moved with keys |
| Size | Alt+F8 | Inactive |
| Minimize | Alt+F9 | Inactive |
| Maximize | Alt+F10 | Opens the associated window (if not already open) and maximizes its size |
| Lower | Alt+F3 | Inactive |
| Close | Alt+F4 | Removes client from mwm management |

To pull down the window menu for the icon box itself, press button 1 with the pointer over the menu button for the icon box. The window menu of the icon box differs from the window menu of a client window: The "Close" selection is replaced with the "PackIcons Shift+Alt+F7" selection. When selected, PackIcons packs the icons in the box to achieve neat rows with no empty slots.

You can also post the window menu by pressing <Shift><Esc> or <Alt><Space>. Pressing <Menu> (the pop-up menu key) causes the icon window menu of the currently selected icon to pop up.

**Input Focus**

mwm supports (by default) a keyboard input focus policy of explicit selection. This means when a window is selected to get keyboard input, it continues to get keyboard input until the window is withdrawn from window management, another window is explicitly selected to get keyboard input, or the window is iconified. Several resources control the input focus. The client window with the keyboard input focus has the active window appearance with a visually distinct window frame.

The following tables summarize the keyboard input focus selection behavior:

| Button Action | Object | Function Description |
|---|---|---|
| Button 1 press | Window / window frame | Keyboard focus selection |
| Button 1 press | Icon | Keyboard focus selection |

| Key Action | Function Description |
|---|---|
| [Alt][Tab] | Move input focus to next window in window stack (available only in explicit focus mode) |
| [Alt][Shift][Tab] | Move input focus to previous window in window stack (available only in explicit focus mode) |

A

**Window Stacking**

There are two types of window stacks: global window stacks and an application's local family window stack.

The global stacking order of windows may be changed as a result of setting the keyboard input focus, iconifying a window, or performing a window manager window stacking function. When keyboard focus policy is explicit the default value of the focusAutoRaise resource is True. This causes a window to be raised to the top of the stack when it receives input focus, for example, by pressing button 1 on the title bar. The key actions defined in the previous table will thus raise the window receiving focus to the top of the stack.

In pointer mode, the default value of focusAutoRaise is False, that is, the window stacking order is not changed when a window receives keyboard input focus. The following key actions can be used to cycle through the global window stack.

| Key Action | Function Description |
|---|---|
| [Alt][ESC] | Place top window on bottom of stack |
| [Alt][Shift][ESC] | Place bottom window on top of stack |

By default, a window's icon is placed on the bottom of the stack when the window is iconified; however, the default can be changed by the lowerOnIconify resource.

Transient windows (secondary windows such a dialog boxes) stay above their parent windows by default. However, an application's local family stacking order may be changed to allow a transient window to be placed below its parent top-level window. The following parameters show the modification of the stacking order for the f.lower function.

Lowers the transient window within the family (staying above the parent) and lowers the family in the global window stack. Lowers the transient window within the family (staying above the parent) but does not lower the family in the global window stack. Lowers the window free from its family stack (below the parent), but does not lower the family in the global window stack.

The parameters within and freeFamily can also be used with f.raise and f.raise_lower.

**X Defaults**

mwm is configured from its resource database. This database is built from the following sources. They are listed in order of precedence, low to high:

/usr/lib/X11/app-defaults/Mwm $HOME/Mwm RESOURCE_MANAGER root window property or $HOME/.Xdefaults XENVIRONMENT variable or $HOME/.Xdefaults-host mwm command line options

The file names /usr/lib/X11/app-defaults/Mwm and $HOME/Mwm represent customary locations for these files. The actual location of the system-wide class resource file may depend on the XFILESEARCHPATH environment variable and the current language environment. The actual location of the user-specific class resource file may depend on the XUSERFILESEARCHPATH and XAPPLRESDIR environment variables and the current language environment.

Entries in the resource database may refer to other resource files for specific types of resources. These include files that contain bitmaps, fonts, and mwm specific resources such as menus and behavior specifications (for example, button and key bindings).

Mwm is the resource class name of mwm and mwm is the default resource name used by mwm to look up resources. the -screens command line option specifies resource names, such as "mwm_b+w" and "mwm_color".) In the following discussion of resource specification, "Mwm" and "mwm" (and the aliased mwm resource names) can be used interchangeably, but "mwm" takes precedence over "Mwm".

mwm uses the following types of resources:

Component Appearance Resources:

These resources specify appearance attributes of window manager user interface components. They can be applied to the appearance of window manager menus, feedback windows (for example, the window reconfiguration feedback window), client window frames, and icons.

General Appearance and Behavior Resources:

These resources specify mwm appearance and behavior (for example, window management policies). They are not set separately for different mwm user interface components.

Client Specific Resources:

These mwm resources can be set for a particular client window or class of client windows. They specify client-specific icon and client window frame appearance and behavior.

Resource identifiers can be either a resource name (for example, foreground) or a resource class (for example, Foreground). If the value of a resource is a filename and if the filename is prefixed by "~/", then it is relative to the path contained in the HOME environment variable (generally the user's home directory).

**Component Appearance Resources**

The syntax for specifying component appearance resources that apply to window manager icons, menus, and client window frames is

Mwm*resource_id

For example, Mwm*foreground is used to specify the foreground color for mwm menus, icons, client window frames, and feedback dialogs.

The syntax for specifying component appearance resources that apply to a particular mwm component is

Mwm*[menu|icon|client|feedback]*resource_id

If menu is specified, the resource is applied only to mwm menus; if icon is specified, the resource is applied to icons; and if client is specified, the resource is applied to client window frames. For example, Mwm*icon*foreground is used to specify the foreground color for mwm icons, Mwm*menu*foreground specifies the foreground color for mwm menus, and Mwm*client*foreground is used to specify the foreground color for mwm client window frames.

The appearance of the title area of a client window frame (including window management buttons) can be separately configured. The syntax for configuring the title area of a client window frame is

Mwm*client*title*resource_id

For example, Mwm*client*title*foreground specifies the foreground color for the title area. Defaults for title area resources are based on the values of the corresponding client window frame resources.

The appearance of menus can be configured based on the name of the menu. The syntax for specifying menu appearance by name is

Mwm*menu*menu_name*resource_id

For example, Mwm*menu*my_menu*foreground specifies the foreground color for the menu named my_menu. The user can also specify resources for window manager menu components, that is, the gadgets which comprise the menu. These may include for example, a menu title, title separator, one or more buttons, and separators. If a menu contains more than one instance of a class, such as multiple PushButtonGadgets, the name of the first instance is "PushButtonGadget1", the second is "PushButtonGadget2", and so on. The following list identifies the naming convention used for window manager menu components: Menu Title LabelGadget - "TitleName" Menu Title SeparatorGadget - "TitleSeparator" CascadeButtonGadget - "CascadeButtonGadget<n>" PushButtonGadget - "PushButtonGadget<n>" SeparatorGadget - "SeparatorGadget<n>"

Refer to the man page for each class for a list of resources which can be specified.

The following component appearance resources that apply to all window manager parts can be specified:

| Component Appearance Resources - All Window Manager Parts | | | |
|---|---|---|---|
| **Name** | **Class** | **Value Type** | **Default** |
| background | Background | color | varies† |
| backgroundPixmap | BackgroundPixmap | string†† | varies† |
| bottomShadowColor | Foreground | color | varies† |
| bottomShadowPixmap | BottomShadowPixmap | string†† | varies† |
| fontList | FontList | string††† | "fixed" |
| foreground | Foreground | color | varies† |
| saveUnder | SaveUnder | T/F | F |
| topShadowColor | Background | color | varies† |
| topShadowPixmap | TopShadowPixmap | string†† | varies† |

†The default is chosen based on the visual type of the screen. ††Image name. See XmInstallImage(3X). †††X11 X Logical Font Description This resource specifies the background color. Any legal X color may be specified. The default value is chosen based on the visual type of the screen.

This resource specifies the background Pixmap of the mwm decoration when the window is inactive (does not have the keyboard focus). The default value is chosen based on the visual type of the screen. This resource specifies the bottom shadow color. This color is used for the lower and right bevels of the window manager decoration. Any legal X color may be specified. The default value is chosen based on the visual type of the screen. This resource specifies the bottom shadow Pixmap. This Pixmap is used for the lower and right bevels of the window manager decoration. The default is chosen based on the visual type of the screen. This resource specifies the font used in the window manager decoration. The character encoding of the font should match the character encoding of the strings that are used. The default is "fixed." This resource specifies the foreground color. The default is chosen based on the visual type of the screen. This is used to indicate whether "save unders" are used for mwm components. For this to have any effect, save unders must be implemented by the X server. If save unders are implemented, the X server saves the contents of windows obscured by windows that have the save under attribute set. If the saveUnder resource is True, mwm will set the save under attribute on the window manager frame of any client that has it set. If saveUnder is False, save unders will not be used on any window manager frames. The default value is False. This resource specifies the top shadow color. This color is used for the upper and left bevels of the window manager decoration. The default is chosen based on the visual type of the screen. This resource specifies the top shadow Pixmap. This Pixmap is used for the upper and left bevels of the window manager decoration. The default is chosen based on the visual type of the screen.

The following component appearance resources that apply to frame and icons can be specified:

| Frame and Icon Components | | | |
|---|---|---|---|
| **Name** | **Class** | **Value Type** | **Default** |
| activeBackground | Background | color | varies† |
| activeBackgroundPixmap | BackgroundPixmap | string†† | varies† |
| activeBottomShadowColor | Foreground | color | varies† |
| activeBottomShadowPixmap | BottomShadowPixmap | string†† | varies† |
| activeForeground | Foreground | color | varies† |
| activeTopShadowColor | Background | color | varies† |
| activeTopShadowPixmap | TopShadowPixmap | string†† | varies† |

†The default is chosen based on the visual type of the screen. ††See XmInstallImage(3X).

This resource specifies the background color of the mwm decoration when the window is active (has the keyboard focus). The default is chosen based on the visual type of the screen. This resource specifies the background Pixmap of the mwm decoration when the window is active (has the keyboard focus). The default is chosen based on the visual type of the screen. This resource specifies the bottom shadow color of the mwm decoration when the window is active (has the keyboard focus). The default is chosen based on the visual type of the screen. This resource specifies the bottom shadow Pixmap of the mwm decoration when the window is active (has the keyboard focus). The default is chosen based on the visual type of the screen. This resource specifies the foreground color of the mwm decoration when the window is active (has the keyboard focus). The default is chosen based on the visual type of the screen. This resource specifies the top shadow color of the mwm decoration when the window is active (has the keyboard focus). The default is chosen based on the visual type of the screen. This resource specifies the top shadow Pixmap of the mwm decoration when the window is active (has the keyboard focus). The default is chosen based on the visual type of the screen.

**General Appearance and Behavior Resources**

The syntax for specifying general appearance and behavior resources is

Mwm*resource_id

For example, Mwm*keyboardFocusPolicy specifies the window manager policy for setting the keyboard focus to a particular client window.

The following general appearance and behavior resources can be specified:

A

| General Appearance and Behavior Resources | | | |
|---|---|---|---|
| **Name** | **Class** | **Value Type** | **Default** |
| autoKeyFocus | AutoKeyFocus | T/F | T |
| autoRaiseDelay | AutoRaiseDelay | millisec | 500 |
| bitmapDirectory | BitmapDirectory | directory | /usr/include/\ X11/bitmaps |
| buttonBindings | ButtonBindings | string | "DefaultBut\ ton-Bindings" |
| cleanText | CleanText | T/F | T |
| clientAutoPlace | ClientAutoPlace | T/F | T |
| colormapFocusPolicy | ColormapFocusPolicy | string | keyboard |
| configFile | ConfigFile | file | .mwmrc |
| deiconifyKeyFocus | DeiconifyKeyFocus | T/F | T |
| doubleClickTime | DoubleClickTime | millisec. | multi-click time |
| enableWarp | enableWarp | T/F | T |
| enforceKeyFocus | EnforceKeyFocus | T/F | T |
| fadeNormalIcon | FadeNormalIcon | T/F | F |
| feedbackGeometry | FeedbackGeometry | string | center on screen |
| frameBorderWidth | FrameBorderWidth | pixels | varies |
| iconAutoPlace | IconAutoPlace | T/F | T |
| iconBoxGeometry | IconBoxGeometry | string | 6x1+0-0 |
| iconBoxName | IconBoxName | string | iconbox |
| iconBoxSBDisplayPolicy | IconBoxSBDisplayPolicy | string | all |
| iconBoxTitle | IconBoxTitle | XmString | Icons |
| iconClick | IconClick | T/F | T |
| iconDecoration | IconDecoration | string | varies |
| iconImageMaximum | IconImageMaximum | wxh | 50x50 |
| iconImageMinimum | IconImageMinimum | wxh | 16x16 |
| iconPlacement | IconPlacement | string | left bottom |
| iconPlacementMargin | IconPlacementMargin | pixels | varies |
| interactivePlacement | InteractivePlacement | T/F | F |
| keyBindings | KeyBindings | string | "DefaultKey\ Bind-ings" |
| keyboardFocusPolicy | KeyboardFocusPolicy | string | explicit |
| limitResize | LimitResize | T/F | T |
| lowerOnIconify | LowerOnIconify | T/F | T |
| maximumMaximumSize | MaximumMaximumSize | wxh (pixels) | 2X screen w&h |
| moveThreshold | MoveThreshold | pixels | 4 |
| moveOpaque | MoveOpaque | T/F | F |
| multiScreen | MultiScreen | T/F | F |
| passButtons | PassButtons | T/F | F |
| passSelectButton | PassSelectButton | T/F | T |
| positionIsFrame | PositionIsFrame | T/F | T |
| positionOnScreen | PositionOnScreen | T/F | T |
| quitTimeout | QuitTimeout | millisec. | 1000 |
| raiseKeyFocus | RaiseKeyFocus | T/F | F |
| resizeBorderWidth | ResizeBorderWidth | pixels | varies |

| | | | |
|---|---|---|---|
| resizeCursors | ResizeCursors | T/F | T |
| screens | Screens | string | varies |
| showFeedback | ShowFeedback | string | all |
| startupKeyFocus | StartupKeyFocus | T/F | T |
| transientDecoration | TransientDecoration | string | menu title |
| transientFunctions | TransientFunctions | string | -minimize -maximize |
| useIconBox | UseIconBox | T/F | F |
| wMenuButtonClick | WMenuButtonClick | T/F | T |
| wMenuButtonClick2 | WMenuButtonClick2 | T/F | T |

This resource is available only when the keyboard input focus policy is explicit. If autoKeyFocus is given a value of True, then when a window with the keyboard input focus is withdrawn from window management or is iconified, the focus is set to the previous window that had the focus. If the value given is False, there is no automatic setting of the keyboard input focus. It is recommended that both autoKeyFocus and startupKeyFocus be True to work with tear off menus. The default value is True. This resource is available only when the focusAutoRaise resource is True and the keyboard focus policy is pointer. The autoRaiseDelay resource specifies the amount of time (in milliseconds) that mwm will wait before raising a window after it gets the keyboard focus. The default value of this resource is 500 (ms). This resource identifies a directory to be searched for bitmaps referenced by mwm resources. This directory is searched if a bitmap is specified without an absolute pathname. The default value for this resource is /usr/include/X11/bitmaps. The directory /usr/include/X11/bitmaps represents the customary locations for this directory. The actual location of this directory may vary on some systems. If the bitmap is not found in the specified directory, XBMLANGPATH is searched. This resource identifies the set of button bindings for window management functions. The named set of button bindings is specified in the mwm resource description file. These button bindings are merged with the built-in default bindings. The default value for this resource is "DefaultButtonBindings". This resource controls the display of window manager text in the client title and feedback windows. If the default value of True is used, the text is drawn with a clear (no stipple) background. This makes text easier to read on monochrome systems where a backgroundPixmap is specified. Only the stippling in the area immediately around the text is cleared. If False, the text is drawn directly on top of the existing background. This resource determines the position of a window when the window has not been given a program- or user-specified position. With a value of True, windows are positioned with the top left corners of the frames offset horizontally and vertically. A value of False causes the currently configured position of the window to be used. In either case, mwm will attempt to place the windows totally on-screen. The default value is True. This resource indicates the colormap focus policy that is to be used. If the resource value is explicit, a colormap selection action is done on a client window to set the colormap focus to that window. If the value is pointer, the client window containing the pointer has the colormap focus. If the value is keyboard, the client window that has the keyboard input focus has the colormap focus. The default value for this resource is keyboard. The resource value is the pathname for an mwm resource description file.

If the pathname begins with "~/", mwm considers it to be relative to the user's home directory (as specified by the HOME environment variable). If the LANG environment variable is set, mwm looks for $HOME/$LANG/configFile. If that file does not exist or if LANG is not set, mwm looks for $HOME/configFile.

If the configFile pathname does not begin with ~/, mwm considers it to be relative to the current working directory.

If the configFile resource is not specified or if that file does not exist, mwm uses several default paths to find a configuration file. If the LANG environment variable is set, mwm looks for the configuration file first in $HOME/$LANG/.mwmrc. If that file does not exist or if LANG is not set, mwm looks for $HOME/.mwmrc. If that file does not exist and if LANG is set, mwm next looks for the file system.mwmrc in the $LANG subdirectory of an implementation-dependent directory. (The default for this directory, if not changed by the implementation, is /usr/lib/X11.) If that file does not exist or if LANG is not set, mwm looks for the file system.mwmrc in the same implementation-dependent directory. This resource applies only when the keyboard input focus policy is explicit. If a value of True is used, a window receives the keyboard input focus when it is normalized (deiconified). True is the default value. This resource is used to set the maximum time (in ms) between the clicks (button presses) that make up a double-click. The default value of this resource is the display's multi-click time. The default value of this resource, True, causes mwm to warp the pointer to the center of the selected window during keyboard-controlled resize and move operations. Setting the value to False causes mwm to leave the pointer at its original place on the screen, unless the user explicitly moves it with the cursor keys or pointing device. If this resource is given a value of True, the keyboard input focus is always explicitly set to selected windows even if there is an indication that they are "globally active" input windows. (An example of a globally active window is a scroll bar that can be operated without setting the focus to that client.) If the resource is False, the keyboard input focus is not explicitly set to globally active windows. The default value is True. If this resource is given a value of True, an icon is grayed out whenever it has been normalized (its window has been opened). The default value is False. This resource sets the position of the move and resize feedback window. If this resource is not specified, the default is to place the feedback window at the center of the screen. The value of the resource is a standard window geometry string with the following syntax:

A

    [=]{+-}xoffset{+-}yoffset]

This resource specifies the width (in pixels) of a client window frame border without resize handles. The border width includes the 3-D shadows. The default value is based on the size and resolution of the screen. This resource indicates whether the window manager arranges icons in a particular area of the screen or places each icon where the window was when it was iconified. The value True indicates that icons are arranged in a particular area of the screen, determined by the iconPlacement resource. The value False indicates that an icon is placed at the location of the window when it is iconified. The default is True. This resource indicates the initial position and size of the icon box. The value of the resource is a standard window geometry string with the following syntax:

    [=][widthxheight][{+-}xoffset{+-}yoffset]

If the offsets are not provided, the iconPlacement policy is used to determine the initial placement. The units for width and height are columns and rows.

The actual screen size of the icon box window depends on the iconImageMaximum (size) and iconDecoration resources. The default value for size is (6 * iconWidth + padding) wide by (1 * iconHeight + padding) high. The default value of the location is +0 -0. This resource specifies the name that is used to look up icon box resources. The default name is "iconbox". This resource specifies the scroll bar display policy of the window manager in the icon box. The resource has three possible values: all, vertical, and horizontal. The default value, "all", causes both vertical and horizontal scroll bars always to appear. The value "vertical" causes a single vertical scroll bar to appear in the icon box and sets the orientation of the icon box to horizontal (regardless of the iconBoxGeometry specification). The value "horizontal" causes a single horizontal scroll bar to appear in the icon box and sets the orientation of the icon box to vertical (regardless of the iconBoxGeometry specification). This resource specifies the name that is used in the title area of the icon box frame. The default value is "Icons". When this resource is given the value of True, the system menu is posted and left posted when an icon is clicked. The default value is True. This resource specifies the general icon decoration. The resource value is label (only the label part is displayed) or image (only the image part is displayed) or label image (both the label and image parts are displayed). A value of activelabel can also be specified to get a label (not truncated to the width of the icon) when the icon is selected. The default icon decoration for icon box icons is that each icon has a label part and an image part (label image). The default icon decoration for stand alone icons is that each icon has an active label part, a label part, and an image part (activelabel label image). This resource specifies the maximum size of the icon image. The resource value is widthxheight (for example, 64x64). The maximum supported size is 128x128. The default value of this resource is 50x50. This resource specifies the minimum size of the icon image. The resource value is widthxheight (for example, 32x50). The minimum supported size is 16x16. The default value of this resource is 16x16. This resource specifies the icon placement scheme to be used. The resource value has the following syntax:

primary_layout secondary_layout [tight]

The layout values are one of the following:

| Value | Description |
|---|---|
| top | Lay the icons out top to bottom. |
| bottom | Lay the icons out bottom to top. |
| left | Lay the icons out left to right. |
| right | Lay the icons out right to left. |

A horizontal (vertical) layout value should not be used for both the primary_layout and the secondary_layout (for example, don't use top for the primary_layout and bottom for the secondary_layout). The primary_layout indicates whether, when an icon placement is done, the icon is placed in a row or a column and the direction of placement. The secondary_layout indicates where to place new rows or columns. For example, top right indicates that icons should be placed top to bottom on the screen and that columns should be added from right to left on the screen. The default placement is left bottom (icons are placed left to right on the screen, with the first row on the bottom of the screen, and new rows added from the bottom of the screen to the top of the screen). A tight value places icons with zero spacing in between icons. This value is useful for aesthetic reasons, as well as X-terminals with small screens. This resource sets the distance between the edge of the screen and the icons that are placed along the edge of the screen. The value should be greater than or equal to 0. A default value (see below) is used if the value specified is invalid. The default value for this resource is equal to the space between icons as they are placed on the screen (this space is based on maximizing the number of icons in each row and column). This resource controls the initial placement of new windows on the screen. If the value is True, the pointer shape changes before a new window is placed on the screen to indicate to the user that a position should be selected for the upper-left hand corner of the window. If the value is False, windows are placed according to the initial window configuration attributes. The default value of this resource is False. This resource identifies the set of key bindings for window management functions. If specified, these key bindings replace the built-in default bindings. The named set of key bindings is specified in mwm resource description file. The default value for this resource is "DefaultKeyBindings". If set to pointer, the keyboard focus policy is to have the keyboard focus set to the client window that contains the pointer (the pointer could also be in the client window decoration that mwm adds). If set to explicit, the policy is to have the keyboard focus set to a client window when the user presses button 1 with the pointer on the client window or any part of the associated mwm decoration. The default value for this resource is explicit. If this resource is True, the user is not allowed to resize a window to greater than the maximum size. The default value for this resource is True. If this resource is given the default value of True, a window's icon appears on the bottom of the window stack when the window is minimized (iconified). A value of False places the icon in the stacking order at the same place as its associated window. The default value of this resource is True. This resource is used to limit the maximum size of a client window as set by the user or client. The resource value is widthxheight (for example, 1024x1024) where the width and height are in pixels. The default value of this resource is twice the screen width and height. This resource controls whether the actual window is moved or a rectangular outline of the window is moved. A default value of False displays a rectangular outline on moves. This resource is used to control the sensitivity of dragging operations that move windows and icons. The value of this resource is the number of pixels that the locator is moved with a button down before the move operation is initiated. This is used to prevent window/icon movement when you click or double-click and there is unintentional pointer movement with the button down. The default value of this resource is 4 (pixels). This resource, if True, causes mwm to manage all the screens on the display. If False, mwm manages only a single screen. The default value is False. This resource indicates whether or not button press events are passed to clients after they are used to do a window manager function in the client context. If the resource value is False, the button press is not passed to the client. If the value is True, the button press is passed to the client window. The window manager function is done in either case. The default value for this resource is False. This resource indicates whether or not to pass the select button press events to clients after they are used to do a window manager function in the client context. If the resource value is False, then the button press will not be passed to the client. If the value is True, the button press is passed to the client window. The window manager function is done in either case. The default value for this resource is True. This resource indicates how client window position information (from the WM_NORMAL_HINTS property and from configuration requests) is to be interpreted. If the resource value is True, the information is interpreted as the position of the MWM client window frame. If the value is False, it is interpreted as being the position of the client area of the window. The default value of this resource is True. This resource is used to indicate that windows should initially be placed (if possible) so that they are not clipped by the edge of the screen (if the resource value is True). If a window is larger than the size of the screen, at least the upper-left corner of the window is on-screen. If the resource value is False, windows are placed in the requested position even if totally off-screen. The default value of this resource is True. This resource specifies the amount of time (in milliseconds) that mwm will wait for a client to update the WM_COMMAND property after mwm has sent the WM_SAVE_YOURSELF message. The default value of this resource is 1000 (ms). (Refer to the f.kill function description for additional information.) This resource is available only when the keyboard input focus policy is explicit. When set to True, this resource specifies that a window raised by means of the f.normalize_and_raise function also receives the input focus. The default value of this resource is False. This resource specifies the width (in pixels) of a client window frame border with resize handles. The specified border width includes the 3-D shadows. The default value is based on the size and resolution of the screen. This is used to indicate whether the resize cursors are always displayed when the pointer is in the window size border. If True, the cursors are shown, otherwise the window manager cursor is shown. The default value is True. This resource specifies the resource names to use for the screens managed by mwm. If mwm is managing a single screen, only the first name in the list is used. If mwm is managing multiple screens, the names are assigned to the screens in order, starting with screen 0. Screen 0 gets the first name, screen 1 the second name, and so on. The default screen names are 0, 1, and so on. This resource controls whether or not feedback windows or confirmation dialogs are displayed. A feedback window shows a client window's initial placement and shows position and size during move and resize operations. Confirmation dialogs can be displayed for certain operations.

A

The value for this resource is a list of names of the feedback options to be enabled or disabled; the names must be separated by a space. If an option is preceded by a minus sign, that option is excluded from the list. The sign of the first item in the list determines the initial set of options. If the sign of the first option is minus, mwm assumes all options are present and starts subtracting from that set. If the sign of the first decoration is plus (or not specified), mwm starts with no options and builds up a list from the resource.

The names of the feedback options are shown below:

| **Name** | **Description** |
|---|---|
| all | Show all feedback (Default value) |
| behavior | Confirm behavior switch |
| kill | Confirm on receipt of KILL signal |
| move | Show position during move |
| none | Show no feedback |
| placement | Show position and size during initial placement |
| quit | Confirm quitting mwm |
| resize | Show size during resize |
| restart | Confirm mwm restart |

The following command line illustrates the syntax for showFeedback:

Mwm*showFeedback: placement resize behavior restart

This resource specification provides feedback for initial client placement and resize, and enables the dialog boxes to confirm the restart and set behavior functions. It disables feedback for the move function. The default value for this resource is all. This resource is available only when the keyboard input focus policy is explicit. When given the default value of True, a window gets the keyboard input focus when the window is mapped (that is, initially managed by the window manager). It is recommended that both autoKeyFocus and startupKeyFocus be True to work with tear off menus. The default value is True. This controls the amount of decoration that mwm puts on transient windows. The decoration specification is exactly the same as for the clientDecoration (client specific) resource. Transient windows are identified by the WM_TRANSIENT_FOR property, which is added by the client to indicate a relatively temporary window. The default value for this resource is menu title (that is, transient windows have frame borders and a titlebar with a window menu button).

An application can also specify which decorations mwm should apply to its windows. If it does so, mwm applies only those decorations indicated by both the application and the transientDecoration resource. Otherwise, mwm applies the decorations indicated by the transientDecoration resource. For more information see the description of XmNmwmDecorations on the VendorShell(3X) reference page. This resource is used to indicate which window management functions are applicable (or not applicable) to transient windows. The function specification is exactly the same as for the clientFunctions (client specific) resource. The default value for this resource is -minimize -maximize.

An application can also specify which functions mwm should apply to its windows. If it does so, mwm applies only those functions indicated by both the application and the transientFunctions resource. Otherwise, mwm applies the functions indicated by the transientFunctions resource. For more information see the description of XmNmwmFunctions on the VendorShell(3X) reference page. If this resource is given a value of True, icons are placed in an icon box. When an icon box is not used, the icons are placed on the root window (default value). This resource indicates whether a click of the mouse when the pointer is over the window menu button posts and leaves posted the window menu. If the value given this resource is True, the menu remains posted. True is the default value for this resource. When this resource is given the default value of True, a double-click action on the window menu button does an f.kill function.

**Client Specific Resources**

The syntax for specifying client specific resources is

Mwm*client_name_or_class*resource_id

For example, Mwm*mterm*windowMenu is used to specify the window menu to be used with mterm clients. The syntax for specifying client specific resources for all classes of clients is

Mwm*resource_id

Specific client specifications take precedence over the specifications for all clients. For example, Mwm*windowMenu is used to specify the window menu to be used for all classes of clients that don't have a window menu specified.

The syntax for specifying resource values for windows that have an unknown name and class (that is, windows that do not have a WM_CLASS property associated with them) is

Mwm*defaults*resource_id

For example, Mwm*defaults*iconImage is used to specify the icon image to be used for windows that have an unknown name and class.

The following client specific resources can be specified:

A

| Client Specific Resources | | | |
|---|---|---|---|
| **Name** | **Class** | **Value Type** | **Default** |
| clientDecoration | ClientDecoration | string | all |
| clientFunctions | ClientFunctions | string | all |
| focusAutoRaise | FocusAutoRaise | T/F | varies |
| iconImage | IconImage | pathname | (image) |
| iconImageBackground | Background | color | icon background |
| iconImageBottomShadowColor | Foreground | color | icon bottom shadow |
| iconImageBottomShadowPixmap | BottomShadow- Pixmap | color | icon bottom shadow pixmap |
| iconImageForeground | Foreground | color | varies |
| iconImageTopShadowColor | Background | color | icon top shadow color |
| iconImageTopShadowPixmap | TopShadow- Pixmap | color | icon top shadow pixmap |
| matteBackground | Background | color | background |
| matteBottomShadowColor | Foreground | color | bottom shadow color |
| matteBottomShadowPixmap | BottomShadow- Pixmap | color | bottom shadow pixmap |
| matteForeground | Foreground | color | foreground |
| matteTopShadowColor | Background | color | top shadow color |
| matteTopShadowPixmap | TopShadow- Pixmap | color | top shadow pixmap |
| matteWidth | MatteWidth | pixels | 0 |
| maximumClientSize | MaximumClientSize | wxh vertical horizontal | fill the screen |
| useClientIcon | UseClientIcon | T/F | F |
| usePPosition | UsePPosition | string | nonzero |
| windowMenu | WindowMenu | string | "DefaultWindow-Menu" |

This resource controls the amount of window frame decoration. The resource is specified as a list of decorations to specify their inclusion in the frame. If a decoration is preceded by a minus sign, that decoration is excluded from the frame. The sign of the first item in the list determines the initial amount of decoration. If the sign of the first decoration is minus, mwm assumes all decorations are present and starts subtracting from that set. If the sign of the first decoration is plus (or not specified), then mwm starts with no decoration and builds up a list from the resource.

An application can also specify which decorations mwm should apply to its windows. If it does so, mwm applies only those decorations indicated by both the application and the clientDecoration resource. Otherwise, mwm applies the decorations indicated by the clientDecoration resource. For more information see the description of XmNmwmDecorations on the VendorShell(3X) reference page.

| Name | Description |
|---|---|
| all | Include all decorations (default value) |
| border | Window border |
| maximize | Maximize button (includes title bar) |
| minimize | Minimize button (includes title bar) |
| none | No decorations |
| resizeh | Border resize handles (includes border) |
| menu | Window menu button (includes title bar) |
| title | Title bar (includes border) |

Examples:

Mwm*XClock.clientDecoration: -resizeh -maximize

This removes the resize handles and maximize button from XClock windows.

Mwm*XClock.clientDecoration: menu minimize border

This does the same thing as above. Note that either menu or minimize implies title. This resource is used to indicate which mwm functions are applicable (or not applicable) to the client window. The value for the resource is a list of functions. If the first function in the list has a minus sign in front of it, then mwm starts with all functions and subtracts from that set. If the first function in the list has a plus sign in front of it, then mwm starts with no functions and builds up a list. Each function in the list must be preceded by the appropriate plus or minus sign and separated from the next function by a space.

An application can also specify which functions mwm should apply to its windows. If it does so, mwm applies only those functions indicated by both the application and the clientFunctions resource. Otherwise, mwm applies the functions indicated by the clientFunctions resource. For more information see the description of XmNmwmFunctions on the VendorShell(3X) reference page.

The table below lists the functions available for this resource:

| Name | Description |
|---|---|
| all | Include all functions (default value) |
| none | No functions |
| resize | f.resize |
| move | f.move |
| minimize | f.minimize |
| maximize | f.maximize |
| close | f.kill |

A

When the value of this resource is True, clients are raised when they get the keyboard input focus. If the value is False, the stacking of windows on the display is not changed when a window gets the keyboard input focus. The default value is True when the keyboardFocusPolicy is explicit and False when the keyboardFocusPolicy is pointer. This resource can be used to specify an icon image for a client (for example, "Mwm*myclock*iconImage"). The resource value is a pathname for a bitmap file. The value of the (client specific) useClientIcon resource is used to determine whether or not user supplied icon images are used instead of client supplied icon images. The default value is to display a built-in window manager icon image. This resource specifies the background color of the icon image that is displayed in the image part of an icon. The default value of this resource is the icon background color (that is, specified by "Mwm*background or Mwm*icon*background). This resource specifies the bottom shadow color of the icon image that is displayed in the image part of an icon. The default value of this resource is the icon bottom shadow color (that is, specified by Mwm*icon*bottomShadowColor). This resource specifies the bottom shadow Pixmap of the icon image that is displayed in the image part of an icon. The default value of this resource is the icon bottom shadow Pixmap (that is, specified by Mwm*icon*bottomShadowPixmap). This resource specifies the foreground color of the icon image that is displayed in the image part of an icon. The default value of this resource varies depending on the icon background. This resource specifies the top shadow color of the icon image that is displayed in the image part of an icon. The default value of this resource is the icon top shadow color (that is, specified by Mwm*icon*topShadowColor). This resource specifies the top shadow Pixmap of the icon image that is displayed in the image part of an icon. The default value of this resource is the icon top shadow pixmap (that is, specified by Mwm*icon*topShadowPixmap). This resource specifies the background color of the matte, when matteWidth is positive. The default value of this resource is the client background color (that is, specified by "Mwm*background or Mwm*client*background). This resource specifies the bottom shadow color of the matte, when matteWidth is positive. The default value of this resource is the client bottom shadow color (that is, specified by "Mwm*bottomShadowColor or Mwm*client*bottomShadowColor). This resource specifies the bottom shadow Pixmap of the matte, when matteWidth is positive. The default value of this resource is the client bottom shadow pixmap (that is, specified by "Mwm*bottomShadowPixmap or Mwm*client*bottomShadowPixmap). This resource specifies the foreground color of the matte, when matteWidth is positive. The default value of this resource is the client foreground color (that is, specified by "Mwm*foreground or Mwm*client*foreground). This resource specifies the top shadow color of the matte, when matteWidth is positive. The default value of this resource is the client top shadow color (that is, specified by "Mwm*topShadowColor or Mwm*client*topShadowColor). This resource specifies the top shadow pixmap of the matte, when matteWidth is positive. The default value of this resource is the client top shadow pixmap (that is, specified by "Mwm*topShadowPixmap or Mwm*client*topShadowPixmap). This resource specifies the width of the optional matte. The default value is 0, which effectively disables the matte. This resource is either a size specification or a direction that indicates how a client window is to be maximized. The resource value can be specified as a size specification widthxheight. The width and height are interpreted in the units that the client uses (for example, for terminal emulators this is generally characters). Alternately, "vertical" or "horizontal" can be specified to indicate the direction in which the client maximizes.

If this resource is not specified, the maximum size from the WM_NORMAL_HINTS property is used if set. Otherwise the default value is the size where the client window with window management borders fills the screen. When the maximum client size is not determined by the maximumClientSize resource, the maximumMaximumSize resource value is used as a constraint on the maximum size. If the value given for this resource is True, a client-supplied icon image takes precedence over a user-supplied icon image. The default value is False, giving the user-supplied icon image higher precedence than the client-supplied icon image. This resource specifies whether Mwm honors program specified position PPosition specified in the WM_NORMAL_HINTS property in the absence of an user specified position. Setting this resource to on, causes mwm to always honor program specified position. Setting this resource to off, causes mwm to always ignore program specified position. Setting this resource to the default value of nonzero cause mwm to honor program specified position other than (0,0). This resource indicates the name of the menu pane that is posted when the window menu is popped up (usually by pressing button 1 on the window menu button on the client window frame). Menu panes are specified in the MWM resource description file. Window menus can be customized on a client class basis by specifying resources of the form Mwm*client_name_or_class*windowMenu (see "Mwm Resource Description File Syntax"). The default value of this resource is "DefaultWindowMenu".

**Resource Description File**

The MWM resource description file is a supplementary resource file that contains resource descriptions that are referred to by entries in the defaults files (.Xdefaults, app-defaults/Mwm). It contains descriptions of resources that are to be used by mwm, and that cannot be easily encoded in the defaults files (a bitmap file is an analogous type of resource description file). A particular mwm resource descriptionfile can be selected using the configFile resource.

The following types of resources can be described in the mwm resource description file: Window manager functions can be bound (associated) with button events. Window manager functions can be bound (associated) with key press events. Menu panes can be used for the window menu and other menus posted with key bindings and button bindings.

**mwm Resource Description File Syntax**

The mwm resource description file is a standard text file that contains items of information separated by blanks, tabs, and newline characters. Blank lines are ignored. Items or characters can be quoted to avoid special interpretation (for example, the comment character can be quoted to prevent it from being interpreted as the comment character). A quoted item can be contained in double quotes ("). Single characters can be quoted by preceding them by the backslash character (\). All text from an unquoted # to the end of the line is regarded as a comment and is not interpreted as part of a resource description. If ! is the first character in a line, the line is regarded as a comment. If a line ends in a backslash character (\), the next line is considered a continuation of that line. Window manager functions can be accessed with button and key bindings, and with window manager menus. Functions are indicated as part of the specifications for button and key binding sets, and menu panes. The function specification has the following syntax:

```
function =          function_name [function_args]
function_name =     window manager function
function_args =     {quoted_item | unquoted_item}
```

The following functions are supported. If a function is specified that isn't one of the supported functions, then it is interpreted by mwm as f.nop. This function causes a beep. This function causes the window or icon that is on the top of the window stack to be put on the bottom of the window stack (so that it no longer obscures any other window or icon). This function affects only those windows and icons that obscure other windows and icons, or that are obscured by other windows and icons. Secondary windows (that is, transient windows) are restacked with their associated primary window. Secondary windows always stay on top of the associated primary window and there can be no other primary windows between the secondary windows and their primary window. If an icon function argument is specified, the function applies only to icons. If a window function argument is specified, the function applies only to windows. This function raises the window or icon on the bottom of the window stack (so that it is not obscured by any other windows). This function affects only those windows and icons that obscure other windows and icons, or that are obscured by other windows and icons. Secondary windows (that is, transient windows) are restacked with their associated primary window. If an icon function argument is specified, the function applies only to icons. If a window function argument is specified, the function applies only to windows. This function causes command to be executed (using the value of the MWMSHELL environment variable if it is set, otherwise the value of the SHELL environment variable if it is set, otherwise /bin/sh). The ! notation can be used in place of the f.exec function name. This function sets the colormap focus to a client window. If this function is done in a root context, the default colormap (set up by the X Window System for the screen where MWM is running) is installed and there is no specific client window colormap focus. This function is treated as f.nop if colormapFocusPolicy is not explicit. This function sets the keyboard input focus to a client window or icon. This function is treated as f.nop if keyboardFocusPolicy is not explicit or the function is executed in a root context. This function is used to terminate a client. If the WM_DELETE_WINDOW protocol is set up, the client is sent a client message event, indicating that the client window should be deleted. If the WM_SAVE_YOURSELF protocol is set up, the client is sent a client message event, indicating that the client needs to prepare to be terminated. If the client does not have the WM_DELETE_WINDOW or WM_SAVE_YOURSELF protocol set up, this function causes a client's X connection to be terminated (usually resulting in termination of the client). Refer to the description of the quitTimeout resource and the WM_PROTOCOLS property. This function lowers a primary window to the bottom of the global window stack (where it obscures no other window) and lowers the secondary window (transient window or dialog box) within the client family. The arguments to this function are mutually exclusive.

The client argument indicates the name or class of a client to lower. If the client argument is not specified, the context that the function was invoked in indicates the window or icon to lower.

Specifying within lowers the secondary window within the family (staying above the parent) but does not lower the client family in the global window stack.

Specifying freeFamily lowers the window to the bottom of the global windows stack from its local family stack. This function causes a client window to be displayed with its maximum size. This function associates a cascading (pull-right) menu with a menu pane entry or a menu with a button or key binding. The menu_name function argument identifies the menu to be used. This function causes a client window to be minimized (iconified). When a window is minimized when no icon box is used, its icon is placed on the bottom of the window stack (so that it obscures no other window). If an icon box is used, the client's icon changes to its iconified form inside the icon box. Secondary windows (that is, transient windows) are minimized with their associated primary window. There is only one icon for a primary window and all its secondary windows. This function causes a client window to be interactively moved. This function installs the next colormap in the list of colormaps for the window with the colormap focus. This function sets the keyboard input focus to the next window/icon in the set of windows/icons managed by the window manager (the ordering of this set is based on the stacking of windows on the screen). This function is treated as f.nop if keyboardFocusPolicy is not explicit. The keyboard input focus is moved only to windows that do not have an associated secondary window that is application modal. If the transient argument is specified, transient (secondary) windows are traversed (otherwise, if only window is specified, traversal is done only to the last focused window in a transient group). If an icon function argument is specified, the function applies only to icons. If a window function argument is specified, the function applies only to windows. This function does nothing. This function causes a client window to be displayed with its normal size. Secondary windows (that is, transient windows) are placed in their normal state along with their associated primary window. This function causes the corresponding client window to be displayed with its normal size and raised to the top of the window stack. Secondary windows (that is, transient windows) are placed in their normal state along with their associated primary window. This function is used to relayout icons (based on the layout policy being used) on the root window or in the icon box. In general this causes icons to be "packed" into the icon grid. This function is used to enable/disable (toggle) processing of key bindings for window manager functions. When it disables key binding processing, all keys are passed on to the window with the keyboard input focus and no window manager functions are invoked. If the f.pass_keys function is invoked with a key binding to disable key-binding processing, the same key binding can be used to enable key-binding processing. This function is used to post the window menu. If a key is used to post the window menu and a window menu button is present, the window menu is automatically placed with its top-left corner at the bottom-left corner of the window menu button for the client window. If no window menu button is present, the window menu is placed at the top-left corner of the client window. This function installs the previous colormap in the list of colormaps for the window with the colormap focus. This function sets the keyboard input focus to the previous window/icon in the set of windows/icons managed by the window manager (the ordering of this set is based on the stacking of windows on the screen). This function is treated as f.nop if keyboardFocusPolicy is not explicit. The keyboard input focus is moved only to windows that do not have an associated secondary window that is application modal. If the transient argument is specified, transient (secondary) windows are traversed (otherwise, if only window is specified, traversal is done only to the last focused window in a transient group). If an icon function argument is specified, the function applies only to icons. If an window function argument is specified, the function applies only to windows. This function terminates mwm (but NOT the X window system). This function raises a primary window to the top of the global window stack (where it is obscured by no other window) and raises the secondary window (transient window or dialog box) within the client family. The arguments to this function are mutually exclusive.

A

The client argument indicates the name or class of a client to lower. If the client is not specified, the context that the function was invoked in indicates the window or icon to lower.

Specifying within raises the secondary window within the family but does not raise the client family in the global window stack.

Specifying freeFamily raises the window to the top of its local family stack and raises the family to the top of the global window stack. This function raises a primary window to the top of the global window stack if it is partially obscured by another window; otherwise, it lowers the window to the bottom of the window stack. The arguments to this function are mutually exclusive.

Specifying within raises a secondary window within the family (staying above the parent window), if it is partially obscured by another window in the application's family; otherwise, it lowers the window to the bottom of the family stack. It has no effect on the global window stacking order.

Specifying freeFamily raises the window to the top of its local family stack, if obscured by another window, and raises the family to the top of the global window stack; otherwise, it lowers the window to the bottom of its local family stack and lowers the family to the bottom of the global window stack. This function causes all windows to be redrawn. This function causes a client window to be redrawn. This function causes a client window to be interactively resized. This function restores the previous state of an icon's associated window. If a maximized window is iconified, then f.restore restores it to its maximized state. If a normal window is iconified, then f.restore restores it to its normalized state. This function restores the previous state of an icon's associated window and raises the window to the top of the window stack. If a maximized window is iconified, then f.restore_and_raise restores it to its maximized state and raises it to the top of the window stack. If a normal window is iconified, then f.restore_and_raise restores it to its normalized state and raises it to the top of the window stack. This function causes mwm to be restarted (effectively terminated and re-executed). screen_number]" This function causes the pointer to be warp to a specific screen number or to the next, previous, or last visited (back) screen. The arguments to this function are mutually exclusive.

The screen_number argument indicates the screen number that the pointer is to be warped. Screens are numbered starting from screen 0.

Specifying next cause the pointer to warp to the next managed screen (skipping over any unmanaged screens).

Specifying prev cause the pointer to warp to the previous managed screen (skipping over any unmanaged screens).

Specifying back cause the pointer to warp to the last visited screen. This function sends a client message of the type _MOTIF_WM_MESSAGES with the message_type indicated by the message_number function argument. The client message is sent only if message_number is included in the client's _MOTIF_WM_MESSAGES property. A menu item label is grayed out if the menu item is used to do f.send_msg of a message that is not included in the client's _MOTIF_WM_MESSAGES property. This function causes a menu separator to be put in the menu pane at the specified location (the label is ignored). This function causes the window manager to restart with the default behavior (if a custom behavior is configured) or revert to the custom behavior. By default this is bound to Shift Ctrl Meta <Key>!. This function inserts a title in the menu pane at the specified location.

A

Each function may be constrained as to which resource types can specify the function (for example, menu pane) and also what context the function can be used in (for example, the function is done to the selected client window). Function contexts are No client window or icon has been selected as an object for the function. A client window has been selected as an object for the function. This includes the window's title bar and frame. Some functions are applied only when the window is in its normalized state (for example, f.maximize) or its maximized state (for example, f.normalize). An icon has been selected as an object for the function.

If a function's context has been specified as icon|window and the function is invoked in an icon box, the function applies to the icon box, not to the icons inside.

If a function is specified in a type of resource where it is not supported or is invoked in a context that does not apply, the function is treated as f.nop. The following table indicates the resource types and function contexts in which window manager functions apply.

| Function | Contexts | Resources |
|---|---|---|
| f.beep | root, icon, window | button, key, menu |
| f.circle_down | root, icon, window | button, key, menu |
| f.circle_up | root, icon, window | button, key, menu |
| f.exec | root, icon, window | button, key, menu |
| f.focus_color | root, icon, window | button, key, menu |
| f.focus_key | root, icon, window | button, key, menu |
| f.kill | icon, window | button, key, menu |
| f.lower | icon, window | button, key, menu |
| f.maximize | icon, window(normal) | button, key, menu |
| f.menu | root, icon, window | button, key, menu |
| f.minimize | window | button, key, menu |
| f.move | icon, window | button, key, menu |
| f.next_cmap | root, icon, window | button, key, menu |
| f.next_key | root, icon, window | button, key, menu |
| f.nop | root, icon, window | button, key, menu |
| f.normalize | icon, window(maximized) | button, key, menu |
| f.normalize_and_raise | icon, window | button, key, menu |
| f.pack_icons | root, icon, window | button, key, menu |
| f.pass_keys | root, icon, window | button, key, menu |
| f.post_wmenu | root, icon, window | button, key |
| f.prev_cmap | root, icon, window | button, key, menu |
| f.prev_key | root, icon, window | button, key, menu |
| f.quit_mwm | root, icon, window | button, key, menu (root only) |
| f.raise | icon, window | button, key, menu |
| f.raise_lower | icon, window | button, key, menu |
| f.refresh | root, icon, window | button, key, menu |
| f.refresh_win | window | button, key, menu |
| f.resize | window | button, key, menu |
| f.restore | icon, window | button, key, menu |
| f.restore_and_raise | icon, window | button, key, menu |
| f.restart | root, icon, window | button, key, menu (root only) |
| f.screen | root, icon, window | button, key, menu |
| f.send_msg | icon, window | button, key, menu |
| f.separator | root, icon, window | menu |
| f.set_behavior | root, icon, window | button, key, menu |
| f.title | root, icon, window | menu |

**Window Manager Event Specification**

Events are indicated as part of the specifications for button and key-binding sets, and menu panes.

Button events have the following syntax:

```
button =          [modifier_list]<button_event_name>
modifier_list =   modifier_name {modifier_name}
```

All modifiers specified are interpreted as being exclusive (this means that only the specified modifiers can be present when the button event occurs). The following table indicates the values that can be used for modifier_name. The [Alt] key is frequently labeled [Extend] or [Meta]. Alt and Meta can be used interchangeably in event specification.

A

| *Modifier* | *Description* |
|---|---|
| Ctrl | Control Key |
| Shift | Shift Key |
| Alt | Alt/Meta Key |
| Meta | Meta/Alt Key |
| Lock | Lock Key |
| Mod1 | Modifier1 |
| Mod2 | Modifier2 |
| Mod3 | Modifier3 |
| Mod4 | Modifier4 |
| Mod5 | Modifier5 |

The following table indicates the values that can be used for button_event_name.

| *Button* | *Description* |
|---|---|
| Btn1Down | Button 1 Press |
| Btn1Up | Button 1 Release |
| Btn1Click | Button 1 Press and Release |
| Btn1Click2 | Button 1 Double-Click |
| Btn2Down | Button 2 Press |
| Btn2Up | Button 2 Release |
| Btn2Click | Button 2 Press and Release |
| Btn2Click2 | Button 2 Double-Click |
| Btn3Down | Button 3 Press |
| Btn3Up | Button 3 Release |
| Btn3Click | Button 3 Press and Release |
| Btn3Click2 | Button 3 Double-Click |
| Btn4Down | Button 4 Press |
| Btn4Up | Button 4 Release |
| Btn4Click | Button 4 Press and Release |
| Btn4Click2 | Button 4 Double-Click |
| Btn5Down | Button 5 Press |
| Btn5Up | Button 5 Release |
| Btn5Click | Button 5 Press and Release |
| Btn5Click2 | Button 5 Double-Click |

Key events that are used by the window manager for menu mnemonics and for binding to window manager functions are single key presses; key releases are ignored. Key events have the following syntax:

```
key =             [modifier_list]<Key>key_name
modifier_list =   modifier_name {modifier_name}
```

All modifiers specified are interpreted as being exclusive (this means that only the specified modifiers can be present when the key event occurs). Modifiers for keys are the same as those that apply to buttons. The key_name is an X11 keysym name. Keysym names can be found in the keysymdef.h file (remove the XK_ prefix).

**Button Bindings**

The buttonBindings resource value is the name of a set of button bindings that are used to configure window manager behavior. A window manager function can be done when a button press occurs with the pointer over a framed client window, an icon, or the root window. The context for indicating where the button press applies is also the context for invoking the window manager function when the button press is done (significant for functions that are context sensitive).

The button binding syntax is

```
Buttons bindings_set_name {
  button   context   function
  button   context   function
                .
                .
  button   context   function }
```

The syntax for the context specification is

```
context =    object[ | context]
object =     root | icon | window | title | frame | border | app
```

The context specification indicates where the pointer must be for the button binding to be effective. For example, a context of window indicates that the pointer must be over a client window or window management frame for the button binding to be effective. The frame context is for the window management frame around a client window (including the border and titlebar), the border context is for the border part of the window management frame (not including the titlebar), the title context is for the title area of the window management frame, and the app context is for the application window (not including the window management frame).

If an f.nop function is specified for a button binding, the button binding is not done.

**Key Bindings**

The keyBindings resource value is the name of a set of key bindings that are used to configure window manager behavior. A window manager function can be done when a particular key is pressed. The context in which the key binding applies is indicated in the key binding specification. The valid contexts are the same as those that apply to button bindings.

The key binding syntax is

```
Keys bindings_set_name {
  key   context   function
  key   context   function
           .
           .
  key   context   function }
```

If an f.nop function is specified for a key binding, the key binding is not done. If an f.post_wmenu or f.menu function is bound to a key, mwm will automatically use the same key for removing the menu from the screen after it has been popped up.

The context specification syntax is the same as for button bindings. For key bindings, the frame, title, border, and app contexts are equivalent to the window context. The context for a key event is the window or icon that has the keyboard input focus (root if no window or icon has the keyboard input focus).

**Menu Panes**

Menus can be popped up using the f.post_wmenu and f.menu window manager functions. The context for window manager functions that are done from a menu is root, icon or window depending on how the menu was popped up. In the case of the window menu or menus popped up with a key binding, the location of the keyboard input focus indicates the context. For menus popped up using a button binding, the context of the button binding is the context of the menu.

The menu pane specification syntax is

```
Menu menu_name {
  label [mnemonic] [accelerator]   function
  label [mnemonic] [accelerator]   function
          .
          .
  label [mnemonic] [accelerator]   function }
```

Each line in the Menu specification identifies the label for a menu item and the function to be done if the menu item is selected. Optionally a menu button mnemonic and a menu button keyboard accelerator may be specified. Mnemonics are functional only when the menu is posted and keyboard traversal applies.

The label may be a string or a bitmap file. The label specification has the following syntax:

```
label =          text | bitmap_file
bitmap_file =    @file_name
text =           quoted_item | unquoted_item
```

The string encoding for labels must be compatible with the menu font that is used. Labels are greyed out for menu items that do the f.nop function or an invalid function or a function that doesn't apply in the current context.

A

A mnemonic specification has the following syntax

```
mnemonic =   _character
```

The first matching character in the label is underlined. If there is no matching character in the label, no mnemonic is registered with the window manager for that label. Although the character must exactly match a character in the label, the mnemonic does not execute if any modifier (such as Shift) is pressed with the character key.

The accelerator specification is a key event specification with the same syntax as is used for key bindings to window manager functions.

### Environment

mwm uses the environment variable HOME specifying the user's home directory.

mwm uses the environment variable LANG specifying the user's choice of language for the mwm message catalog and the mwm resource description file.

mwm uses the environment variables XFILESEARCHPATH, XUSERFILESEARCHPATH, XAPPLRESDIR, XENVIRONMENT, LANG, and HOME in determining search paths for resource defaults files. mwm may also use XBMLANGPATH to search for bitmap files.

mwm reads the $HOME/.motifbind file if it exists to install a virtual key bindings property on the root window. For more information on the content of the .motifbind file, see VirtualBindings(3X).

mwm uses the environment variable MWMSHELL (or SHELL, if MWMSHELL is not set), specifying the shell to use when executing commands via the f.exec function.

### Files

/usr/lib/X11/$LANG/system.mwmrc /usr/lib/X11/system.mwmrc /usr/lib/X11/app-defaults/Mwm $HOME/Mwm $HOME/.Xdefaults $HOME/$LANG/.mwmrc $HOME/.mwmrc $HOME/.motifbind

## RELATED INFORMATION

VendorShell(3X), VirtualBindings(3X), X(1), and XmInstallImage(3X).

## NAME

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

## SYNOPSIS

**resize** [-c | -u] [-h | -x | -s [row col]]

## DESCRIPTION

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

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

After resizing the window, the user would type:

```
% rs
```

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

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

## OPTIONS

The following options may be used with *resize*:

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

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

**-h** This option indicates that *resize* should use Hewlett Packard terminal escape sequences to obtain the terminal's new window size.

**-x** This option indicates that *resize* should use VT102 escape sequences to obtain the terminal's new window size.

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

## FILES

/etc/termcap for the base termcap entry to modify.
$HOME/.profile *sh*(1), *ksh*(1), and *keysh*(1) user's functions for *resize.*
~/.cshrc user's alias for the command.

## SEE ALSO

sh(1), ksh(1), csh(1), keysh(1), eval(1), hpterm(1), tset(1), xterm(1)

## AUTHORS

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

## BUGS

If the *-s* option is used, it must be the last option specified.

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

## NAME

rgb - X Window System color database creator.

## SYNOPSIS

**rgb** [*filename*] [<*input filename*]

## DESCRIPTION

*rgb* creates a data base used by the X window system server for its colors. Stdin is used as its input and must be in the format of:

```
0-255 0-255 0-255    colorname
```

A

For example:

```
  0    0    0    black
  0  128    0    green
255  255  255    white
```

*rgb* stands for *r*ed-*g*reen-*b*lue. Each element can have no intensity (0) to full intensity (255). How the elements are combined determines the actual color. The name given to the color can be descriptive or fanciful.

In other words, the sequence:

```
  0    0  128
```

can be given the name 'blue' or 'unicorn blue'. There can also be two (or more) entries with the same element numbers or names.

## ARGUMENTS

*filename* If filename is given, *rgb* produces two files; *filename.dir* and *filename.pag*. Otherwise the default filename is */usr/lib/X11/rgb*.

## ORIGIN

MIT Distribution

## SEE ALSO

X(1)

**NAME**

sb2xwd - translate Starbase bitmap to xwd bitmap format

**SYNOPSIS**

**sb2xwd**

**DESCRIPTION**

This command translates a bitmap file created by one of the Starbase bitmap-to-file procedures into an *XWD* format file. The *XWD* format is defined by the *xwd(1)* and *xwud(1)* X window dump utility programs. The Starbase bitmap file format is described in *bitmapfile(4)*. Translation is done from standard input to standard output.

Starbase bitmaps created in *pixel-major* format will be translated into *ZPixmap* format xwd bitmaps. *Plane-major full-depth* Starbase bitmaps are translated into *XYPixmap* format xwd bitmaps.

XWD bitmaps produced by *sb2xwd* will be of the *DirectColor* visual class if the Starbase bitmap's colormap mode is *CMAP_FULL*. Other multiplane Starbase bitmaps having colormap modes of *CMAP_NORMAL* or *CMAP_MONOTONIC* will result in *PseudoColor* XWD bitmaps. Single plane Starbase bitmaps are converted to *GreyScale* XWD bitmaps.

**OPTIONS**

none

**EXAMPLES**

```
sb2xwd < sbfile > xwdfile
```

Translates the Starbase image in *sbfile* to XWD format and places the result in *xwdfile.*

```
sb2xwd < sbimage | xwud
```

Translates the image in *sbimage,* piping the results to *xwud* for display.

**RESTRICTIONS**

*sb2xwd* accepts only *full-depth* Starbase bitmaps.

Starbase bitmaps must be 1-8, 12, or 24 planes deep. Bitmaps of depth 1-8 must have either CMAP_NORMAL or CMAP_MONOTONIC colormap modes. Bitmaps of depths 12 or 24 must have the CMAP_FULL colormap mode.

A 12 plane bitmap must be stored in three banks and have a display enable mask of 0x0F or 0xF0.

**ORIGIN**

Hewlett-Packard GTD

**SEE ALSO**

xwd(1), xwud(1), bitmapfile(4).

*Starbase Graphics Techniques, HP-UX Concepts and Tutorials,* chapters on "Color" and "Storing and Printing Images".

## NAME

stconv - Utility to convert scalable type symbol set map formats

## SYNOPSIS

**stconv** *infile* [-hmq] [-d *mapdir*] [-to *ACG|HPMSL|other*]

## DESCRIPTION

A

The *stconv* utility is used to convert scalable typeface symbol set maps (**.sym** files) for Intellifont fonts from one symbol list numbering format to another. The installation default symbol sets map their respective symbols into appropriate HP Master Symbol List character codes (the default format provided in **.ifo** typeface libraries). Since it is possible, however, to use *stload(1M)* to generate **.ifo** libraries that use some other symbol list numbering (such as native Agfa Compugraphic ACG numbering), *stconv* provides a way to modify existing **.sym** files to map characters to non-MSL numbers.

A *symbol list* assigns a unique numeric value to each individual character in the global collection of characters available from a specific typeface manufacturer. The specific numbering scheme used is usually unique to that vendor, and typically identifies hundreds or even thousands of characters. A *symbol set* (or *character set*) is generally a particular subset of characters taken from this master collection, with each character being assigned another unique code in accordance with the some specific registry/encoding scheme (such as ISO8859 Latin-1 or HP Roman-8). In this respect, the **.sym** files used by the HP Scalable Type subsystem are actually *symbol set maps*, equating all the character codes comprising a specific character set registry/encoding to the corresponding character numbers of a particular typeface vendor's symbol list.

Intellifont format scalable typeface libraries are commercially available with HP Master Symbol List character numbering (HPMSL) in products obtained from the HP MasterType Library, and with Agfa Compugraphic symbol list numbering (ACG) for products from the Agfa Typeface Library.

## OPTIONS

There are several command line parameters which are described below.

*infile* Required name of the **.sym** file to be converted.

**-d** *mapdir*

Specifies the name of the directory containing the symbol list conversion map (see below). This directory should contain the file **ACG-HPMSL**, plus any optional additional symbol list maps. If unspecified, *stconv* looks for an appropriate map in */usr/lib/X11/fonts/stadmin/ifo/charsets*. The file **ACG-HPMSL** defines the translation between Agfa Compugraphic symbol list numbering and HP Master Symbol List numbering.

**-h** Requests help.

**-m** Requests that *stconv* list the conversion map.

**-q** Requests that *stconv* should run quietly.

**-to** *format*

Specifies the new symbol list format. The installation-default symbol sets use **HPMSL** format; to generate a symbol set for native Agfa Compugraphic symbol list character codes, you should specify **-to ACG**.

## EXAMPLES

**stconv -to ACG roman8.sym >acgr8.sym**

This reads the HPMSL symbol map **roman8.sym**, and writes the ACG version to **acgr8.sym** via stdout redirection. The new symbol map can subsequently be used to generate an HP Roman-8 font from an Agfa Typeface Library product.

## SYMBOL LIST CONVERSION MAPS

A *symbol list conversion map* defines the translation of character numbers between two different symbol lists. The map is a simple text file contain two columns of character codes: the first column is the first format's character code, the second column is the second format's character code. Lines that begin with anything other than two distinct values are ignored. The name of the conversion must be the names of the two formats concantenated together with a hyphen, with care taken to insure the name before the hyphen reflects the numbering scheme used in the first column, and the name after the hyphen reflects the numbering scheme used in the second column. A single conversion map file is sufficient for converting a symbol list of either format into a symbol list of the opposite format. As an example, the first few lines of the file **ACG-HPMSL** are:

```
# ACG-HPMSL
# First column is ACG number
# Second column is corresponding HPMSL number
1  86
2  81
3  74
4  80
```

## FILES

*Stconv* takes input only from the specified file, and always sends output to **stdout**.

## NOTE

*Stconv* supports only the **.sym** files used for Intellifont scalable fonts, and has no application to other types of scalable fonts (such as Type 1).

## SEE ALSO

stlicense(1M), stmkfont(1), stmkdirs(1), stload(1M)

## COPYRIGHT

## NAME

stlicense - server access control program for X

## SYNOPSIS

**stlicense** [-v] [-fp directory] {-fn typeface | -pr product} [[+-]device ...]

## DESCRIPTION

The *stlicense* program is run interactively by the font administrator to give devices access to typefaces. Responsibility for maintaining security rests with users *root*, *bin* , and the owners of the font directories.

## OPTIONS

A

*stlicense* accepts the command line options described below.

**-fp** *directory*
The path of directories to search for the specified product or typeface. If several directories are specified, they are separated by the colon (":") character. Stlicense will search the path until it finds a directory that contains the the specified product or typeface. It will use that directory for all of its operations. If no **-fp** option is specifies, the value of environment variable **STPATH** is used. If neither **-fp** nor **STPATH** is present, the default path is **/usr/lib/X11/fonts/ifo.st.**

**-fn** *typeface*
The typeface being licensed. It is specified as an XLFD font name. The entire font name does not have to be specified. The program will use the first typeface whose name matches the name specified.

**-pr** *product*
Specifying a product instead of a typeface provides a convenient way to license collections of typefaces. Product names are established when typefaces are loaded. Use either the **-fn** or the **-pr** option, but never both. One of them must be present for license additions or removals.

**-v** The verbose option controls the number messages returned. By default, stlicense prints messages only when a request could not be fulfilled. With the verbose option, status messages are printed for all licensing requests.

**-help** If this is the only option specified, a brief message listing the valid options to *stlicense* will be printed.

+*device* The device is granted a license to the product or typeface specified. Adding a product license to the device means adding licenses to all of the typefaces which comprise the product. Typefaces licensed to device **STSYSTEM** will be available to all devices.

The device is specified in the form *host:device.*
The host name is a node on the network. They can be discless nodes, workstations, and the like. The device, usually a printer, is attached to the host. The name of the device must be a valid file name. Special device DISPLAYS refers to all servers running on the host. Special device PRINTERS refers to all printers connected to the host.

The *host* defaults to the host on which *stlicense* is running.
The device defaults to DISPLAYS. Adding a product license to a device which already has a license for the product causes the device's fonts.dir file to be updated with the current definition of the product.

*-device* The license for specified typeface or product is removed from this device.

*device* The licenses for specified typeface or product are displayed.

*nothing* If no devices are given, the list of devices that have licenses for the typeface is printed on the standard output.

## FILES

This program updates the device's license files in the **licenses** subdirectory of the typeface directory (***.st**). It creates and deletes licensing files as necessary. If files for the device do not exist, it creates them. Deleting the last license from a device will cause the device's licensing files to be deleted.

Hosts have directories in **licenses**. Devices have directories within their host's directory. Typefaces available to the device are stored in **fonts.dir**. Licenses issued to the device are stored in **products.dir**.

The typefaces that comprise valid products are specified in the **products** directory of typeface directories (***.st**).

## EXAMPLES

**stlicense -pr builtin +STSYSTEM:DISPLAYS**

Licenses the Intellifont fonts contained in the product "builtin" (and listed in /usr/lib/X11/fonts/ifo.st/products/builtin) for all displays.

**stlicense -fp /usr/lib/X11/fonts/type1.st -pr builtin +STSYSTEM:DISPLAYS**

Licenses the Type 1 fonts contained in the product "builtin" (and listed in /usr/lib/X11/fonts/type1.st/products/builtin) for all displays.

## SEE ALSO

stload (1M), stmkfont(1), stconv(1M), stmkdirs(1)

## COPYRIGHT

## NAME

stload - Utility to load Scalable Type outlines

## SYNOPSIS

**stload** [options] *directory|filespec...*

## DESCRIPTION

The *stload* utility creates scalable typeface libraries (**.ifo** files) from Agfa Compugraphic Font Access and Interchange Standard (CG/FAIS) data. (An FAIS library on floppy disk or CD-ROM media is the common means of distribution for scalable typeface products that is compatible with the HP Font Server and Scalable Typeface subsystem implementation. It is also is the distribution format used by HP Type Director/DOS and the HP MasterType library of scalable typeface floppy disks.) *Stload* also simplifies the installation of new symbol sets and existing **.ifo** files from an archive repository, creates Tagged Font Metric **.tfm** files, and establishes new product entries in the Scalable Typeface licensing structure. *Stload* should generally be run interactively by the system font administrator to load new scalable typefaces into a Scalable Typeface directory.

A

Several typefaces come bundled with the Scalable Typeface system and are automatically installed with the product; additional typefaces may be acquired separately. Since new typefaces are generally distributed on MS-DOS floppy media, the system administrator may need to use utilities such as *doscp* or *cp* to transfer the typeface files from the distribution media to a temporary location on the target system. CG/FAIS file naming conventions prohibit multiple CG/FAIS diskettes from being copied into a common repository, so you should create a unique directory on the host system to contain each CG/FAIS diskette you load. Once all of the CG/FAIS typeface files have been loaded onto the target system, *stload* is used to compile them into **.ifo** scalable typeface libraries and install the libraries in an appropriate *typefaces* directory.

*Stload* does not grant licensed access to newly-installed typefaces. Font administrators must use the *stlicense* utility to give devices access to new typefaces.

## OPTIONS

There are several command line options which are described below.

*directory|filespec*
: This is a required parameter specifying the name of the directory or filespec of the data to be loaded. You may specify one or more directories containing the FAIS data extracted from a typeface distribution diskette, or one or more repositories containing **.ifo** and/or **.sym** files, or explicit or wildcarded filespecs of existing and/or **.sym** files.

**-d** *mapdir*
: Specifies the directory containing the symbol list map required by the **-to** *format* option. This directory must contain the file **ACG-HPMSL**, plus any optional additional symbol list maps. If left unspecified, *stload* assumes the symbol list maps can be found in */usr/lib/X11/fonts/stadmin/ifo/charsets*.

**-dos** Specifies that the typeface library should be written as a TypeDirector/DOS **.typ** file. Typefaces libraries written in this format cannot be used by the HP-UX Scalable Type subsystem, but have the feature of being expandable -- you can add new typefaces to an existing DOS format library. (See "EXAMPLES" below for more details.)

**-f** *libname*
: Specifies the name of the scalable typeface library into which FAIS data should be loaded. Normally, loading from FAIS data generates a separate **.ifo** or **.typ** library file for each typeface, and each library file's name indicates that particular typeface's ID and character complement code. The **-f** option overrides this automatic naming convention and lets you add typefaces into a single library with any arbitrary name.

**-fp** *path* Specifies the name of the base directory under which *typefaces*, *metrics*, and *products* directories should be used as the target directories for typeface, metric, and product definition files. These directories will be created as necessary if they don't already exist. Any attempt to load typeface, metric, and product definition files elsewhere by using the **-o** option will be ignored. (If neither the **-fp** nor **-o** options are specified, the default base directory is */usr/lib/X11/fonts/ifo.st*. See **-o** below for more option information.)

Note that **-fp** does *not* affect where **.sym** files will be loaded. Unless you use the **-o** option to direct them elsewhere, symbol sets will still be installed into */usr/lib/X11/fonts/stadmin/ifo/charsets*.

**-h** Requests help. **Stload** will print out a command line syntax summary and a list of command line switch options. No other processing will take place.

**-id[,id...]**
Identifies one or more specific typefaces to be loaded from CG/FAIS data. Normally, *stload* installs all of the typefaces it can find, loading each typeface into its own **.ifo** file. If you wish to install only one or two specific faces, you must use this option. (Specific typeface numbers can be determined by using **stload -list** to examine the contents of the FAIS directory; the "Face" column reveals typeface IDs.)

A

**-link** Causes *stload* to install files the output directory by making *hard links* back to the original repository in *directory* rather than by copying the original files. Note that this works only when loading **.sym** files and pre-existing **.ifo** files. All **.ifo** files loaded from FAIS data are created as new files in the output directory.

**-list** Prints a list of the data located in *directory* on the standard output device. The directory may contain either FAIS data, **.ifo** typeface libraries, or **.sym** symbol set files. No other processing takes place.

**-o** *path* Specifies the name of the output directory in which to write the final **.ifo**, **.sym**, and **.tfm** files. If the directory does not exist, *stload* will attempt to create it. If left unspecified, these files will be written to the following default output directories:
write **.ifo** files to */usr/lib/X11/fonts/ifo.st/typefaces*
write **.sym** files to */usr/lib/X11/fonts/stadmin/ifo/charsets*
write **.tfm** files to */usr/lib/X11/fonts/ifo.st/metrics*

**-p** *product-number*
Associates a product number with newly-loaded typefaces (**.ifo** files). Although this can be any arbitrary filename-sized string, it should reflect the product number printed on the typeface distribution package and media. The *stlicense* utility requires this product number in order to license loaded typefaces for use with specific devices. Product numbers are written into a *products* directory at the same directory level as the **.ifo** output directory. For example, if you specified the **-o** *path* option, product numbers will be written to *path/../products*; if you do not specify the any output directory, product numbers will be written to files in */usr/lib/X11/fonts/ifo.st/products*. The *products* directory will be automatically created if it does not already exist.

**-sym** Causes *stload* to install files the output directory by making *symbolic links* back to the original repository in *directory* rather than by copying the original files. Note that this works only for loading **.sym** files and pre-existing **.ifo** files. All **.ifo** files loaded from FAIS data are created as new files in the output directory.

**-tfm** Causes *stload* to update Tagged Font Metric **.tfm** files in the output directory (or in */usr/lib/X11/fonts/ifo.st/metrics* if no output directory is specified).

**-to** *format*
Specifies the symbol list that should be used for assigning character ID codes when loading FAIS data. If left unspecified, character IDs will be assigned in accordance with the HP Master Symbol List (HPMSL); to retain Agfa Compugraphic (native FAIS) ID numbers, you should specify **-to ACG**.

**-u** Specifies that the **.dir** files should *not* be updated. Stload normally automatically uses the *stmkdirs* utility to update the **fonts.dir**, **charsets.dir**, and **metrics.dir** files in the output directory or directories.

**-v** Specifies that **stload** should run verbosely.

## EXAMPLES

**stload -p C2050A#D15 .**
Compiles the CG/FAIS data in the current directory and puts the resulting .ifo files in */usr/lib/X11/fonts/ifo.st/typefaces*. Existing .ifo files in the current directory will also be

copied; existing .sym files will be copied into */usr/lib/X11/fonts/stadmin/ifo/charsets*; appropriate .tfm files will be added to */usr/lib/X11/fonts/ifo.st/metrics*. The product number required for future licensing of these typefaces by **stlicense** is C2050A#D15. **Fonts.dir**, **charsets.dir**, and **metrics.dir** are updated to reflect the additions.

**stload -o . -dos -f big3.ifo -92500 times**
**stload -o . -dos -f big3.ifo -93717 courier**
**stload -o . -p BIG3 -f big3.ifo -92041 univers**

Compiles three different CG/FAIS typefaces into a single .ifo library. The library must be explicitly named to prevent stload from assigning names, and all but the last typeface must be loaded in DOS format to keep the library expandable. The final invocation loads the last typeface, converts the complete DOS library into true .ifo format, and makes the typefaces licensable via the product name BIG3.

A

**stload -list /usr/lib/X11/fonts/ifo.st/typefaces**

Lists all of the typefaces currently contained in */usr/lib/X11/fonts/ifo.st/typefaces.*

## FILES

Remember that *stload* requires write access to the directory that will contain the compiled typeface libraries. This implies that only the system font administrator should be able to add typefaces to the typeface collection in */usr/lib/X11/fonts/ifo.st/typefaces*. The files *fonts.dir*, *charsets.dir*, and *metrics.dir* will be automatically updated to reflect additions. Typeface licenses are not modified.

## RESTRICTIONS

Stload can process a maximum of 20000 glyphs per typeface.

## SEE ALSO

stlicense(1M), stmkfont(1), stmkdirs(1), stconv(1M)

## COPYRIGHT

## NAME

stmkdirs - Utility to build Scalable Type ".dir" and ".tfm" files.

## SYNOPSIS

**stmkdirs** [options] *directory* [*directory* ...]

## DESCRIPTION

The *stmkdirs* utility is used to support Intellifont and Type 1 scalable fonts. It creates the **fonts.dir, charsets.dir**, and **metrics.dir** files used by the Font Server and Scalable Type subsystem.

*Stmkdirs* supports scalable typefaces as follows:

**Intellifont**
: *Stmkdirs* can be used to generate **fonts.dir** files containing entries for Intellifont outline files, **charsets.dir** files cataloging character sets for Intellifont fonts, Tagged Font Metrics (**.tfm**) files describing font metrics for Intellifont outlines, and **metrics.dir** files cataloging **.tfm** files.

**Type 1** *Stmkdirs* can generate **fonts.dir** files containing entries for Type 1 outline files.

A

## OPTIONS

There are several command line options which are described below.

*directory* This specifies the names of one or more directories to be processed. At least one *directory* parameter must be specified. *Stmkdirs* will create whatever *.dir* files are needed in a directory: if a directory contains bitmapped fonts or font outlines, it will create or update **fonts.dir**; if character set files, **charsets.dir**; if TFM files, **metrics.dir.**

**-d** *directory*
: Specifies the directory containing default character set (.sym) files and the symbol list map (such as ACG-HPMSL) required by the **-tfm** *directory* option in order to construct TFM files. If left unspecified, *stmkdirs* assumes the this directory to be */usr/lib/X11/fonts/stadmin/ifo/charsets*. Character set files will be taken from this directory only if none are found in any of the processed directories.

**-sl** *syms* The **-sl** switch informs *stmkdirs* that the font library (.ifo) files being processed conform to some symbol list other than HP Master Symbol List (HPMSL). This information is necessary for generation of correct TFM files. For example, for a font loaded by *stload* with the ACG symbol list, **-sl** ACG should be used in the *stmkdirs* invocation to ensure placing correct information in the TFM files.

**-tfm** *directory*
: The **-tfm** switch instructs *stmkdirs* to build Tagged Font Metrics (TFM) files in the specified directory. TFM files contain font metrics information used by some applications to determine information such as character spacing. Data in the TFM files is gleaned from typeface library **.ifo** files and from character set **.sym** files; TFM files will not be built if *stmkdirs* does not find the required library and character set files. (If *stmkdirs* does not find character set files in any of the directories it processes, it will use the character set files in /usr/lib/X11/fonts/stadmin/ifo/charsets unless pointed elsewhere via the **-d** option). TFM files are generated only for Intellifont outlines.

The default behavior of *stmkdirs* is to attempt to generate complete **fonts.dir, charsets.dir**, and **metrics.dir** files in all directories specified. You can cause certain items to *not* be generated or included in the final files by using "-" suppression options. Similarly, to generate *only* certain items, you should use one or more "+" options.

The command line is processed from left to right, making it possible for you to apply different options to different directories all in the same command. To avoid confusing behavior you should therefore take care to specify all switch options *before* specifying directory names.

Data inclusion and exclusion switches include:

**+c** Requests that **charsets.dir** be generated.

**-c** Requests that **charsets.dir** *not* be generated.

**+f** Requests that **fonts.dir** be generated and should include **bitmap fonts (those normally found by** *mkfontdir*). **This is equivalent to requesting +mo.**

**-f** Requests that **fonts.dir** *not* be generated.

**+m** Requests that **fonts.dir** be generated and should include bitmap fonts (those normally found by *mkfontdir*).

**-m** Requests that bitmap fonts (those normally found by *mkfontdir*) be *excluded* from any newly generated **fonts.dir**.

**+o** Requests that **fonts.dir** be generated and should include

A

**-o** Requests that **.ifo** scalable typeface libraries be *excluded* from any newly generated **fonts.dir**.

**+r** Requests that **metrics.dir** be generated.

**-r** Requests that **metrics.dir** *not* be generated.

Additional switches include:

**-a** Ensures that any symbol set catalogued in **charsets.dir** can be used with any **.ifo** scalable typeface library catalogued in **fonts.dir**. (Normally, the Font Server recognizes only combinations in which the typeface library actually contains all or most of the glyphs specified by the symbol set.)

**-b** Suppress creation of backup files. Normally, whenever *stmkdirs* constructs a new **fonts.dir**, **charsets.dir**, or **metrics.dir** file, any pre-existing old version of that file is renamed (to its original name with a tilde "~" appended) rather than overwritten. The **-b** option disables this feature.

**-h** Print usage information on stdout.

## STMKDIRS VERSUS MKFONTDIR

When generating **fonts.dir** files, *stmkdirs* acts like an extended version of the *mkfontdir(1)* utility. In addition to recognizing the bitmapped font files handled by *mkfontdir*, *stmkdirs* recognizes Intellifont (**.ifo**) and Type 1 (**.pfa** and **.pfb**) font files, and adds entries for them to the *fonts.dir* file.

As with *mkfontdir*, the *fonts.scale* can be used to add entries for file types not recognized by *stmkdirs*. However, *fonts.scale* is not needed for Intellifont and Type 1 scalable fonts.

## EXAMPLES

**stmkdirs -tfm /usr/lib/X11/fonts/ifo.st/metrics /usr/lib/X11/fonts/ifo.st/typefaces**

Requests that **fonts.dir** and, if appropriate, **charsets.dir** files be built in */usr/lib/X11/fonts/ifo.st/typefaces*. In addition, the **-tfm** switch requests that **.tfm** files plus an appropriate **metrics.dir** file be built in */usr/lib/X11/fonts/ifo.st/metrics*.

**stmkdirs /usr/lib/X11/fonts/type1.st/typefaces**

Generates a **fonts.dir** in the /usr/lib/X11/fonts/type1.st/typefaces directories.

## SEE ALSO

stlicense(1M), stmkfont(1), stload(1M)

## COPYRIGHT

Copyright 1990, Hewlett-Packard Company
See *X(1)* for a full statement of rights and permissions.

**NAME**

stmkfont - Scalable Typeface font compiler to create X and PCL fonts

**SYNOPSIS**

**stmkfont** [options] *xlfdname*

**DESCRIPTION**

The *stmkfont* utility is a bit-mapped font generator for creating X and PCL fonts from Intellifont scalable typeface data. By specifying desired font characteristics via an *X Logical Font Description* (XLFD) name, the user can instruct *stmkfont* to generate an almost limitless variety of font flavors from one or more Agfa Compugraphic Intellifont typeface libraries. Possible output formats are **BDF**, and **PCL** for various HP printers.

**REQUIRED PARAMETERS**

*xlfdname*

The last argument on the command line is always assumed to be the *required XLFD name*. The XLFD name is the means by which you specify the typeface, font size, and additional treatments for the final font. (See **xlfd(3)** for information on XLFD construction). The XLFD name should begin with a field-delimiter hyphen and specify from one to fourteen contiguous fields. If less than 14 fields are given, *stmkfont* will automatically append wildcard fields to fill the name out to fourteen. *Stmkfont* then attempts to qualify the XLFD name by looking for a matching scalable typeface descriptor in **fonts.dir**, and upon finding a one, proceeds to build the font.

**SWITCH OPTIONS**

**-C** Output a catalog of typeface/character set combinations. For the given XLFD specification, *stmkfont* will generate a list of fully qualified XLFD names that reflect the various typefaces and character sets that could be used to construct a final font. No font is generated when this option is used.

**-I** Send additional status information to stderr, such as the requested and final XLFD names, return status of the underlying *stmkfont*() call, and a list of characters that either failed font generation or were not in the symbol set.

**-P** Send "one-percent progress dots" to stderr. As *stmkfont* constructs the final font, it will output a stream of period characters at regular time intervals. Exactly 100 periods will be output; when the 100th dot has been sent, the font is ready.

**-S** If one of the PCL output formats is requested, this option causes *stmkfont* to output to stderr the PCL selection string required to select the resulting font on the PCL printer.

**-T** Suppress temporary output file. In its normal mode of operation, *stmkfont* takes time to "dribble" output into a **/tmp** tempfile, and then quickly "burst" copy the complete tempfile to the actual output file or device. While this file re-copy does cause *stmkfont* to take somewhat longer to produce results, it minimizes the amount of time that a parent process must spend "listening" to *stmkfont*'s output. If overall speed is more critical than time spent actually writing the final output, this switch can be used to bypass the tempfile and "dribble" output directly to the final destination.

**-V** Requests that a fully qualified XLFD name be sent to stderr without continuing to generate final output.

**-cf** *CharsetFile*

**-cp** *CharsetPath*

These options allow you to specify the subdirectory (under one of the database directories) and/or filename of a symbol set mapping file. Once *stmkfont* has determined the name of the scalable typeface library it will use, it extracts the filename extension from the library's name, and uses it in conjunction with *CharsetPath*, *CharsetFile*, and the database directory trees *Primary* and *Alternate* to locate an appropriate character set map. *Stmkfont* will look for *CharsetFile* (or **charsets.dir**) in several directories, in the following order:

1. *Primary/TypefaceExtension/CharsetPath*
2. *Alternate/TypefaceExtension/CharsetPath*

3. *{STPATH}/TypefaceExtension/CharsetPath*

If left unspecified, the default *CharsetPath* is **charsets**, and the default *CharsetFile* is determined by scanning the above directories for **charsets.dir**, then using the **Charset Registry** and **Charset Encoding** properties of the XLFD name to extract from it a charset map filename. *Note: In the above directory search hierarchy there should normally exist only one charsets.dir file. Stmkfont will stop searching when the first charsets.dir is encountered. If that charsets.dir does not contain an appropriate registry/encoding to match the XLFD name, stmkfont will be unable to generate the requested font.*

A

In practice, the character sets are usually found in the *Alternate* directory's *CharsetPath* subdirectory. When this product is installed, for example, the character sets are placed in the **/usr/lib/X11/fonts/stadmin/ifo/charsets** directory, where *stmkfont* finds them with the default *Alternate* and *CharsetPath* values.

The *Alternate* directory is also the customary home of the **misc/st.dev** file used by *stmkfont* to determine PCL printer characteristics.

Because the *Alternate* directory is the customary home of many important typeface-independent files, it is usually inadvisable to use the **-d2** option.

**-d1** *Primary*
**-d2** *Alternate*
Whenever *stmkfont* must open a typeface library, a character set map, or a ***.dir** control file, it searches through several directories in a specific order until either the required file is found, or the list of search directories are exhausted. These *database directories* and the order in which they are searched are as follows:

1. *Primary* (default **/usr/lib/X11/fonts/ifo.st**)
2. *Alternate* (default **/usr/lib/X11/fonts/stadmin**)
3. *{STPATH}* (environment variable)

The **-d1** and **-d2** options let you change the *Primary* and *Alternate* database directories. If a requested file cannot be found based on either of these two paths, additional places to look can be specified by using the STPATH environment variable.

In practice, typefaces are usually found in the *Primary* directory, and character sets are found in the *Alternate* directory's *CharsetPath* directory (see below).

**-dv** *device*
Specifies the device for which a font is to be made, in the format *host:device*. If not specified or partly specified (*host:* or *:device*), *host* defaults to the executing machine's hostname, and device defaults to PRINTERS or DISPLAYS (depending on the output format specified). See *stlicense(1)* for further information on the format of the *device* parameter.

**-f** *format*
Specifies the output format for the requested font, which must be one of the following: BDF or one of the supported PCL printer formats. Output formats currently supported are: **LJPLUS**, **LJII**, **LJ2000**, **LJIIP**, **LJIII** (bitmapped fonts for various models of LaserJet), **PJXL** (bitmapped fonts for the PaintJet XL), and **PCLEO** (scalable fonts for the LaserJet III). Bitmapped fonts reflect any transformations (such as emboldening and obliqueing) requested in the *xlfdname* argument; scalable fonts (PCLEO format) do not.

If left unspecified, output format is BDF.

**-h** There is no **-h** option. For online help, run *stmkfont* without any parameters.

**-nf** *fonts.dir*
**-ns** *charsets.dir*
**-nt** *typefaces.dir*

These three options allow you to specify alternate names for the three control files that may be used by *stmkfont*. **fonts.dir** contains a list of typeface library files and the XLFD "outline" name descriptions of those typefaces. **charsets.dir** contains associates character set registry, encoding, and requirements with particular character set map files. **typefaces.dir** assigns "official" names to individual Agfa Compugraphic typeface IDs.

**-nv** *name*
Specifies an alternate environment variable name to use instead of **STPATH**.

**-o** *outfile*
This option specifies the output file. Default is stdout.

**-q** Run quietly (suppress error messages).

**-v** Similar to the -V option, -v requests that a fully qualified XLFD name be sent to stderr before *stmkfont* begins generating glyphs. The string will be terminated with a newline character and stderr will be flushed before any glyph output appears.

**-w** This option causes all bitmap output data to be suppressed, resulting in only header and trailer information being generated. Header metrics will accurately reflect the entire font that would have been emitted, but the character count will show zero.

## ENVIRONMENT

### STPATH

*STPATH* specifies the last-resort paths to be searched when looking for typeface libraries, character set mapping files, and ***.dir** files. Searching *STPATH* is only considered if requested files are not found in the *Primary* or *Alternate* paths (normally, **/usr/lib/X11/fonts/ifo.st** and **/usr/lib/X11/fonts/stadmin**). *STPATH* has a format similar to the *PATH* variable, consisting of one or more paths concatenated together with colons.

## ABOUT XLFD NAMES

If the *xlfdname* argument contains any embedded blanks the entire XLFD name should be enclosed in quotes. Also, while it is good practice to always specify *something* in all XLFD fields, null fields (double dashes) are permissible and will be treated as though they contain asterisk wildcards. Certain fields of the XLFD name will "default" if specified as zero or wild:

```
Point Size (field 8) - default value is 120 (12 point)
X resolution (field 9) - default value is 100 dots per inch (BDF)
                         default value is printer resolution (PCL)
Y resolution (field 10) - default value is 100 dots per inch (BDF)
                          default value is printer resolution (PCL)
```

Resolution of other fields containing wildcards depends wholly on the contents and ordering of data in **fonts.dir** and **charsets.dir**.

## ABOUT THE VARIOUS PCL FORMATS

The PCL bitmapped output formats for the various LaserJet printers are identical. *Stmkfont* uses the format specified with the *-f* option to ensure that the font will work within the restrictions imposed by the target printer. For example, the LJPLUS printer cannot handle the large glyphs allowed by the other models, nor can it handle character sets that define glyphs for the control characters.

## EXAMPLES

To generate a BDF file for a 14-point/110 DPI "cg times" font using the iso8859-1 character set:

```
stmkfont "-agfa-cg times-normal-r-normal-*-*-140-110-110-*-*-iso8859-1"
```

To generate the same font for a LaserJet II:

```
stmkfont -f LJII "-agfa-cg times-normal-r-normal-*-*-140-*-*-*-*-iso8859-1"
```

Note that the resolution fields will default to the printer's 300 DPI resolution. The output will be a LaserJet II font *without* the font management sequences for assigning font ID and for making the font temporary or permanent.

To search for fonts in a directory “$HOME/fonts” containing scalable fonts and a fonts.dir directory:

```
stmkfont -d1 "$HOME/fonts" "-agfa-cg times-normal-r----140-----hp-roman8"
```

Note here that empty fields are simply run together as a string of dashes. While this is an unrecommended departure from the XLFD format standard, *stmkfont* will allow such a specification and treats the empty fields as though they contain asterisk wildcards.

## FILES

A

/usr/lib/X11/fonts/stadmin/ifo/charsets/charsets.dir
/usr/lib/X11/fonts/ifo.st/typefaces/fonts.dir
/usr/lib/X11/fonts/stadmin/misc/st.dev

## NOTE

*Stmkfont* supports only Intellifont scalable fonts; it cannot be used to generate fonts from Type 1 or any other format of scalable outlines.

## ERRORS

**Typeface does not contain an HP alias**
The typeface file does not contain sufficient information to construct a PCL font.

**LJPLUS printer does not support this charset**
A character set was requested that cannot be used on the LaserJet Plus printer. Any character set that defines glyphs for characters 0-31 or 128-159 cannot be used on the LaserJet Plus printer.

**Font too large for printer**
The requested font size is larger than can be handled by the target printer.

## SEE ALSO

mkfontdir(1), stmkdirs(1), stlicense(1M), stload(1M)

## COPYRIGHT

(C) Copyright 1990 Hewlett-Packard Company

## NAME

**vueicon** - The HP VUE Icon Editor

## SYNOPSIS

**vueicon** [options]

## DESCRIPTION

The *Vueicon* program is an icon editor for use within the X Window System environment. The Icon Editor allows you to create new icons, modify existing icons, or create an icon from any visible image on the display. It supports both XPM and XBM format icons, including hotspot. The Icon Editor also supports HP VUE drag and drop, allowing the user to drag file icons from the HP VUE file manager onto the Icon Editor window for editing.

A

## OPTIONS

The following options are available from the command line:

**-f** *file*

This options takes a bitmap or pixmap file name. The Icon Editor will first try to open the exact file name. If this fails, it will look for the same file name with a *.pm* or *.bm* suffix.

**-x** *widthxheight*

This option specifies an initial geometry for the icon image. If a file is specified using the -f option, the size of that icon supersedes the geometry specified.

**-noMessaging**

This option forces the Icon Editor to forego connecting to the BMS. This will result in the Icon Editor being unable to support the dragging and dropping of files. This is equivalent to setting the *useMessaging* resource to False.

**-session** *file*

This option takes the name of a session file as an additional parameter. The icon editor is run with the specified session file name. The session file is a file that was previously saved by the icon editor through a session shutdown. This option causes all other command line options to be ignored, as all settings are taken from the specified session file.

## RESOURCES

The following table and descriptions define resources used to specify data for the *vueicon* program.

**Vueicon Resource Set**

| **Name** | **Class** | **Type** | **Default** |
|---|---|---|---|
| useMessaging | UseMessaging | Boolean | True |

**useMessaging**

Used by the HP VUE Lite configuration. Setting this resource to True is equivalent to specifying the **-noMessaging** command line option.

## COPYRIGHT

Copyright 1991, 1992 Hewlett-Packard Company.

## ORIGIN

Hewlett-Packard Company.

## NAME

**vuelogin – The HP VUE Login Manager.**

## SYNOPSIS

**vuelogin** [-config *configuration_file*] [-daemon] [-debug *debug_level*] [-error *error_log_file*] [-nodaemon] [-resources *resource_file*] [-server *server_entry*] [-session *session_program*]

## DESCRIPTION

A

*Vuelogin* manages a collection of X displays, both local and possibly remote. The emergence of X terminals guided the design of several parts of this system, along with the development of the X Consortium standard XDMCP ( *X Display Manager Control Protocol*). *Vuelogin* provides services similar to those provided by *init*(1M), *getty*(1M) and *login*(1) on character terminals: prompting for login and password, authenticating the user, and running a "session."

A "session" is defined by the lifetime of a particular process; in the traditional character-based terminal world, it is the user's login shell process. In the HP VUE context, it is the HP VUE Session Manager. This is because in a windowing environment, a user's login shell process does not necessarily have any terminal-like interface with which to connect.

If the HP VUE Session Manager is not used, the typical *vuelogin* substitute is either a window manager with an exit option, or a terminal emulator running a shell, where the lifetime of the terminal emulator is the lifetime of the shell process that it is running; thus reducing the X session to an emulation of the character-based terminal session.

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

Because *vuelogin* provides the first interface that users see, it is designed to be simple to use and easy to customize to the needs of a particular site.

## OPTIONS

All options, except **-config**, specify values that can also be specified in the configuration file as resources. Typically, customization is done via the configuration file rather than command line options. The options are most useful for debugging and one-shot tests.

**-config** *configuration_file*
: Specifies a resource file that specifies the remaining configuration parameters. If no file is specified and the file */usr/vue/config/Xconfig* exists, *vuelogin* uses it.

**-daemon**
: Specifies "true" as the value for the **daemonMode** resource. This makes *vuelogin* close all file descriptors, disassociate the controlling terminal and put itself in the background when it first starts up (just like the host of other daemons).

**-debug** *debug_level*
: Specifies the numeric value for the **debugLevel** resource. A non-zero value causes *vuelogin* to print debugging statements to the terminal; it also disables the **daemonMode** resource, forcing *vuelogin* to run synchronously.

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

**-nodaemon**
: Specifies "false" as the value for the **daemonMode** resource.

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

**-server** *server_entry*
: Specifies the value for the **servers** resource. See **servers** below for more detail.

**-udpPort** *port_number*
Specifies the value for the **requestPort** resource. This sets the port-number that *vuelogin* monitors for XDMCP requests. Since XDMCP uses the registered well-known udp port 177, this resource should probably not be changed except for debugging.

**-session** *session_program*
Specifies the value for the **session** resource. This indicates the program to run when the user has logged in as the session.

## CONTROLLING THE SERVER

*Vuelogin* controls local servers using POSIX signals. SIGHUP is expected to reset the server, closing all client connections and performing other clean up duties. SIGTERM is expected to terminate the server. If these signals do not perform the expected actions, the resources **resetSignal** and **termSignal** can specify alternate signals.

A

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

## CONTROLLING VUELOGIN

*Vuelogin* responds to two signals: SIGHUP and SIGTERM. When sent a SIGHUP, *vuelogin* rereads the configuration file and the file specified by the **servers** resource and determines whether entries have been added or removed. If a new entry has been added, *vuelogin* starts a session on the associated display. Entries that have been removed are disabled immediately, meaning that any session in progress is terminated without notice, and no new session is started.

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

## ENVIRONMENT

*Vuelogin* invokes the user's session with the following default environment:

| | |
|---|---|
| DISPLAY | is set to the associated display name |
| EDITOR | is set to /usr/vue/bin/vuepad |
| HOME | is set to the home directory of the user |
| KBD_LANG | is set to the value of LANG for applicable languages |
| LANG | is set to the current NLS language (if any) |
| LC_ALL | is set to the current NLS language (if any) |
| LC_MESSAGES | is set to the current NLS language (if any) |
| LOGNAME | is set to the user name |
| MAIL | is set to /usr/mail/$USER |
| PATH | is set to the value of the **userPath** resource |
| USER | is set to the user name |
| SHELL | is set to the user's default shell (from /etc/passwd) |
| TERM | is set to xterm |
| TZ | is set to the value of the **timeZone** resource or system default |
| XAUTHORITY | may be set to an authority file |

Three methods are available to modify or add to this list depending on the desired scope of the resulting environment variable.

The **environment** resource is available in the *vuelogin* configuration file to allow setting of environment variables on a global or per-display basis. Variables specified by this method are available to both the display's X server process and the user's session and override any default settings. The resource accepts a string of <name> = <value> pairs separated by at least one space or tab. The values specified must be constants because no shell is used to parse the string. See the **Resources** section below for details on setting this resource.

For example:

```
Vuelogin*environment:      SB_DISPLAY_ADDR=0xB00000 \
                           WMSHMSPC=0x200000
```

Note: The environment variables LANG and TZ have their own dedicated resources in the configuration file and should not be set via **environment**.

A

Environment variables that require processing by a shell or are dependent on the value of another environment variable can be specified in the startup script *Xsession*. These variables are loaded into the environment of all users on the display, but not to the X server process. They override any previous settings of the same variable. The *Xsession* script accepts ksh syntax for setting environment variables.

For example.

```
MAIL=/usr/mail/$USER
```

Finally, personal environment variables can be set on a per-user basis in the script file $HOME/*.vueprofile*. *Vuelogin* accepts either sh, ksh, or csh syntax for the commands in this file. The commands should only be those that set environment variables, not any that perform terminal I/O, excepting *tset(1)* or *stty(1)*. If the first line of *.vueprofile* is #!/bin/sh, #!/bin/ksh, or #!/bin/csh, *vuelogin* uses the appropriate shell to parse *.vueprofile*. Otherwise, the user's default shell ($SHELL) is used.

## INTERNATIONALIZATION

All labels and messages are localizable. The message catalog *vuelogin.cat* contains the localized representations of the default labels and messages. *Vuelogin* reads the appropriate message catalog indicated by the LANG environment variable and displays the localized strings. An option on the authentication screen allows the user to override the default language for the subsequent session. If the authentication screen has been localized for the selected language, it is redisplayed in that language; otherwise, it is displayed in the default language. In either case, the LANG environment variable is set appropriately for the resulting session.

The resource **language** is available in the *vuelogin* configuration file to change the default language for a display.

The resource **languagelist** is also available in the *vuelogin* configuration file to override the default set of languages displayed on the authentication screen.

## RESOURCES

The actions of *vuelogin* can be controlled through the use of various configuration files, which are in the X resource format. Some resources control the behavior of *vuelogin* in general, some can be specified for a particular display, and others control the appearance of the authentication screen. The general and display-specific resources are specified in the configuration file named by the **-config** command line option. All resources should be prepended with the application name **Vuelogin**.

**Vuelogin General Resource Set**

| Name | Class | Type | Default |
|---|---|---|---|
| accessFile | AccessFile | String | NULL |
| autoRescan | AutoRescan | Boolean | True |
| daemonMode | DaemonMode | Boolean | False |
| debugLevel | DebugLevel | Int | 0 |

| | | | |
|---|---|---|---|
| errorLogFile | ErrorLogFile | String | NULL |
| keyFile | KeyFile | String | /usr/vue/config/Xkeys |
| lockPidFile | LockPidFile | Boolean | True |
| pidFile | PidFile | String | NULL |
| authDir | AuthDir | String | /usr/vue/config |
| removeDomainname | RemoveDomainname | Boolean | True |
| requestPort | RequestPort | Int | 177 |
| servers | Servers | String | :0 Local local /usr/bin/X11/X :0 |
| sysParmsFile | SysParmsFile | String | /etc/src.sh |
| timeZone | TimeZone | String | MST7MDT |
| wakeupInterval | WakeupInterval | Int | 10 |

A

The *vuelogin* general resources are not display-specific and are applied to all displays where appropriate.

**accessFile**
To prevent unauthorized XDMCP service this file contains a database of hostnames which are allowed direct access to this machine. The format of this file is described in the section **Xdmcp Access Control.**

**authDir**
This is a directory name that *vuelogin* uses to temporarily store authorization files for displays using XDMCP.

**autoRescan**
This boolean controls whether *vuelogin* rescans the configuration file and server file after a session terminates and the files have changed. You can force *vuelogin* to reread these files by sending a SIGHUP to the main process.

**daemonMode**
*Vuelogin* can make itself into an unassociated daemon process. This is accomplished by forking and leaving the parent process to exit, then closing file descriptors and releasing the controlling terminal. This is inconvenient when attempting to debug *vuelogin.* Setting this resource to "false" disables **daemonMode.**

If *vuelogin* is started from /etc/inittab, it should not be run in daemon mode. Otherwise the *init* process will think it has terminated and will attempt to restart it.

**debugLevel**
A non-zero value specified for this integer resource enables debugging information to be printed. It also disables daemon mode, which redirects the information into the bit-bucket. Specifying a non-zero debug level also allows non-root users to run *vuelogin,* which is not normally useful.

**errorLogFile**
Error output is normally directed at the system console. To redirect it, set this resource to any file name. This file also contains any output directed to stderr by *Xstartup, Xsession* and *Xreset*, so it contains descriptions of problems in those scripts as well.

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

**lockPidFile**
This is the resource that controls whether *vuelogin* uses file locking to prevent multiple logins.

**pidFile** The filename specified is created to contain an ASCII representation of the process-id of the main *vuelogin* process. This is quite useful when reinitializing the system. *Vuelogin* also uses file locking to attempt to prevent more than one daemon running on the same machine.

**removeDomainname**
When computing the display name for XDMCP clients, the resolver typically creates a fully qualified host name for the terminal. As this is sometimes confusing, *vuelogin* removes the domain name portion of the host name if it is the same as the domain name for the local host when this variable is set.

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

A

**servers** This resource either specifies a file name full of server entries, one per line (if the value starts with a slash), or a single server entry. Each entry indicates a display that should constantly be managed and that is not using XDMCP. Each entry consists of at least three parts: a display name, a display class, a display type, and (for local servers) a command line to start the server. A typical entry for local display number 0 is:

```
:0 Local local@console /usr/bin/X11/X :0
```

The display types are:

| | |
|---|---|
| local | a local display, i.e. one that has a server program to run |
| foreign | a remote display, i.e. one that has no server program to run |

The display name must be something that can be passed in the **-display** option to any X program. This string is used in the display-specific resources to specify the particular display, so be careful to match the names (e.g., use ":0 local /usr/bin/X11/X :0" instead of "localhost:0 local /usr/bin/X11/X :0" if your other resources are specified as "Vuelogin._0.session"). The display class portion is also used in the display-specific resources as the class portion of the resource. This is useful if you have a large collection of similar displays (a group of X terminals, for example) and want to set resources for groups of them. When using XDMCP, the display is required to specify the display class, so perhaps your X terminal documentation describes a reasonably standard display class string for your device.

On local bitmaps, the user may choose a "No Windows" option via the login screen, which temporarily suspends the X-server and presents the traditional character "login:" prompt. The user can then log in and perform non-X related tasks. When the user finishes and logs out, the X-server is restarted, and the login screen is redisplayed.

In order to support "No Windows" mode, the display must have an associated Internal Terminal Emulator (ITE) device. By default, *vuelogin* associates the ITE device "console" (/dev/console) with display ":0". If your configuration does not match this default, specify "@<device>" for the display(s) with an associated ITE and "@none" for all other displays listed in the **servers** file.

**sysParmsFile**
This resource specifies a file containing shell commands, one of which sets the timezone environment variable (TZ) for the system. If the timezone is set via the shell syntax, "TZ=", *vuelogin* can use this information to set the timezone for the user session.

**timeZone**
This resource specifies the local time zone for *vuelogin*. It is loaded into the environment of *vuelogin* as the value of the variable **TZ** and inherited by all subsequent sessions.

Some systems maintain a configuration file that contains the timezone setting (ex. /etc/src.sh). See the resource **sysParmsFile**.

**wakeupInterval**
If the user selects "No Windows" mode from the login screen, *vuelogin* terminates the X-

server and allows the traditional character-based login prompt, "login:" to become visible. If the user does not log in within 2 * **wakeupInterval** seconds, the X-server is restarted. Once the user has logged in, *vuelogin* checks every **wakeupInterval** seconds to see if the user has logged out. If so, the X-server is restarted and the login screen is redisplayed.

**Vuelogin Display Resource Set**

| **Name** | **Class** | **Type** | **Default** |
|---|---|---|---|
| authorize | Authorize | Boolean | False |
| authName | AuthName | String | MIT-MAGIC-COOKIE-1 |
| authFile | AuthFile | String | /usr/vue/config/auth-server |
| cpp | Cpp | String | /lib/cpp |
| environment | Environment | String | NULL |
| failsafeClient | FailsafeClient | String | /usr/bin/X11/xterm |
| grabServer | GrabServer | Boolean | True |
| grabTimeout | GrabTimeout | Int | 3 sec. |
| language | Language | String | NULL |
| languageList | LanguageList | String | NULL |
| openDelay | OpenDelay | Int | 5 sec. |
| openRepeat | OpenRepeat | Int | 5 sec. |
| openTimeout | OpenTimeout | Int | 30 sec. |
| pingInterval | PingInterval | Int | 5 min. |
| pingTimeout | PingTimeout | Int | 5 min. |
| reset | Reset | String | NULL |
| resetForAuth | ResetForAuth | Boolean | False |
| resetSignal | Signal | Int | 1 (SIGHUP) |
| resources | Resources | String | NULL |
| session | Session | String | NULL |
| startAttempts | StartAttempts | Int | 4 |
| startup | Startup | String | NULL |
| systemPath | SystemPath | String | /usr/bin/X11:/bin:/usr/bin:/etc |
| systemShell | SystemShell | String | /bin/sh |
| terminateServer | TerminateServer | Boolean | False |
| termSignal | Signal | Int | 15 (SIGTERM) |
| userAuthDir | UserAuthDir | String | /tmp |
| userPath | UserPath | String | /usr/bin/X11:/bin:/usr/bin:/usr/contrib/bin:/usr/local/bin |
| vuelite | Vuelite | Boolean | False |
| xrdb | Xrdb | String | /usr/bin/X11/xrdb |

A

*Vuelogin* display resources can be specified for all displays or for a particular display. To specify a particular display, the display name is inserted into the resource name between "Vuelogin" and the final resource name segment. For example, **Vuelogin.expo_0.startup** is the name of the resource defining the startup shell file on the "expo:0" display. The resource manager separates the name of the resource from its value with colons, and separates resource name parts with dots, so *vuelogin* uses underscores for the dots and colons when generating the resource name.

Resources can also be specified for a class of displays by inserting the class name instead of a display name. A display that is not managed by XDMCP can have its class affiliation specified in the file referenced by the **servers** resource. A display using XDMCP supplies its class affiliation as part of the XDMCP packet.

**authorize**

**authorize** is a boolean resource that controls whether *vuelogin* generates and uses authorization for the server connections. (See **authName.**)

**authName**
If **authorize** is used, **authName** specifies the type of authorization to be used. Currently, *vuelogin* supports only MIT-MAGIC-COOKIE-1 authorization, XDM-AUTHORIZATION-1 could be supported, but DES is not generally distributable. XDMCP connections state which authorization types are supported dynamically, so **authName** is ignored in this case. (See **authorize.**)

**authFile**
This file is used to communicate the authorization data from *vuelogin* to the server, using the *-auth* server command line option. It should be kept in a write-protected directory to prevent its erasure, which would disable the authorization mechanism in the server.

A

**cpp** This specifies the name of the C preprocessor that is used by xrdb.

**environment**
This resource can contain a set of <name>=<value> pairs separated by a space or tab. Each item is loaded into the environment of the server and session. See the **Environment** section for details.

**failsafeClient**
If the default session fails to execute, *vuelogin* falls back to this program. This program is executed with no arguments, but executes using the same environment variables as the session would have had. (See **The Xsession File** below.)

**grabServer** (See **grabTimeout.**)

**grabTimeout**
To improve security, *vuelogin* grabs the server and keyboard while reading the name and password. The **grabServer** resource specifies if the server should be held while the name and password is read. When FALSE, the server is ungrabbed after the keyboard grab succeeds; otherwise, the server is grabbed until just before the session begins. The **grabTimeout** resource specifies the maximum time *vuelogin* will wait for the grab to succeed. The grab may fail if some other client has the server grabbed, or possibly if the network latencies are very high. The **grabTimeout** resource has a default of 3 seconds; be cautious when using this resource, since a user can be deceived by a look-alike window on the display. If the grab fails, *vuelogin* kills and restarts the server (if possible) and session.

Some X-terminals cannot display their configuration screens while the server is grabbed. Setting **grabServer** to false will allow the screens to be displayed, but opens the possibility that a user's login name can be stolen by copying the contents of the login screen. Since the keyboard is still grabbed and the password is not echoed, the password cannot be stolen.

**language**
This resource specifies the default setting for the **LANG** environment variable. If the *vuelogin* screen is localized for that language, it is displayed appropriately; otherwise, it is displayed in the language "C". The user may temporarily override this setting via an option on the login screen. When the subsequent session terminates, the **LANG** variable reverts to this setting.

**languageList**
This resource allows the user to override the default set of languages displayed in the "Language" menu of the login screen. It is useful if the set of languages actually used on a particular display is smaller than the set installed on the system. The resource value is a list of valid values for the **LANG** environment variable. Language values should be separated by one or more spaces or tabs.

**openDelay** (See **startAttempts.**)

**openRepeat** (See **startAttempts.**)

**openTimeout** (See **startAttempts.**)

**pingInterval** (See **pingTimeout.**)

**pingTimeout**
To discover when remote displays disappear, *vuelogin* occasionally "pings" them, using an X connection and sending XSync requests. **pingInterval** specifies the time (in minutes) between successive ping attempts, and **pingTimeout** specifies the maximum wait time (in minutes) for the terminal to respond to the request. If the terminal does not respond, the session is terminated. *Vuelogin* does not ping local displays. Although it may seem harmless, it is undesirable when the workstation session is terminated as a result of the server hanging for NFS service and not responding to the ping.

**reset** This specifies a program that is run (as root) after the session terminates. By default no program is run. The conventional name is *Xreset*. See **The Xreset File** below.

**resetForAuth**
The original implementation of authorization in the sample server reread the authorization file at server reset time, instead of when checking the initial connection. Since *vuelogin* generates the authorization information just before connecting to the display, an old server does not get current authorization information. This resource causes *vuelogin* to send SIGHUP to the server after setting up the file, causing an additional server reset to occur, during which time the new authorization information is read.

**resetSignal**
This resource specifies the signal *vuelogin* sends to reset the server. See the section **Controlling The Server**

**resources**
This resource specifies the name of the file to be loaded by *xrdb* (*1*) as the resource database onto the root window of screen 0 of the display. This resource data base is loaded just before the authentication procedure is started, so it can control the appearance of the "login" window. See the section below on the authentication screen, which describes the various resources that are appropriate to place in this file. There is no default value for this resource, but the conventional name is *Xresources*. See **Authentication Screen Resources** below.

**session** This specifies the session to be executed (not running as root). By default, */usr/bin/X11/xterm* is run. The conventional name is *Xsession*. See **The Xsession File** below.

**startAttempts**
Four numeric resources control the behavior of *vuelogin* when attempting to open reluctant servers: **openDelay**, **openRepeat**, **openTimeout**, and **startAttempts**. **openDelay** is the duration (in seconds) between successive attempts; **openRepeat** is the number of attempts to make; **openTimeout** is the amount of time to wait while actually attempting the opening (i.e., the maximum time spent in the *connect* (2) syscall); and **startAttempts** is the number of times the entire process occurs before giving up on the server. After **openRepeat** attempts have been made, or if **openTimeout** seconds elapse in any particular attempt, *vuelogin* terminates and restarts the server, attempting to connect again. This process is repeated **startAttempts** time, at which point the display is declared dead and disabled. (See **openDelay**, **openRepeat**, and **openTimeout**.)

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

**systemPath**
*Vuelogin* sets the PATH environment variable for the startup and reset scripts to the value of this resource. Note the conspicuous absence of "." from this entry. This is a good practice to follow for root; it avoids many system penetration schemes.

**systemShell**
*Vuelogin* sets the SHELL environment variable for the startup and reset scripts to the value of this resource.

**terminateServer**
This boolean resource specifies whether the X server should be terminated when a

session terminates (instead of resetting it). This option can be used if the server tends to grow without bound over time in order to limit the amount of time the server is run continuously.

**termSignal**
This resource specifies the signal *vuelogin* sends to terminate the server. See the section **Controlling The Server**

**userAuthDir**
When *vuelogin* cannot write to the usual user authorization file ($HOME/.Xauthority), it creates a unique file name in this directory and points the environment variable XAUTHORITY at the created file.

A

**userPath**
*Vuelogin* sets the PATH environment variable for the session to this value. It should be a colon-separated list of directories; see *sh(1)* for a full description.

**vuelite** Setting this resource to "True" restricts the display to only allowing fail-safe or VUE Lite sessions. The "HP VUE Session" selection is disabled.

**xrdb** Specifies the program used to load the resources.

## AUTHENTICATION SCREEN RESOURCES

The authentication screen reads a name-password pair from the keyboard. As this is a Motif toolkit client, colors, fonts and some layout options can be controlled with resources. Resources for this screen should be put into the file named by the **resources** resource.

The default logo on the authentication screen may be replaced with a bitmap of the user's choice. The following resources are available in addition to the standard Motif set in order to control positioning of the logo and the drop shadow. The resources should be prefaced with the string **Vuelogin*logo*** when specified.

**Logo Resource Set**

| Name | Class | Type | Default |
|---|---|---|---|
| bitmapFile | BitmapFile | String | NULL |
| dropShadowBackground | DropShadowBackground | Pixel | dynamic |
| dropShadowForeground | DropShadowForeground | Pixel | dynamic |
| dropShadowBackgroundPixmap | DropShadowBackgroundPixmap | String | dynamic |
| dropShadowThickness | DropShadowThickness | Int | dynamic |
| verticalOffset | VerticalOffset | Int | dynamic |
| x | X | Position | -1 |
| y | X | Position | -1 |

**bitmapFile**
Specifies the absolute path name to the bitmap file to be used for the logo.

**dropShadowBackground**
Specifes the background color for the drop shadow.

**dropShadowForeground**
Specifes the foreground color for the drop shadow.

**dropShadowBackgroundPixmap**
Specifes the pixmap to be used for the drop shadow. This can either be a built-in Motif pixmap or the absolute path name to a bitmap to be used as the tile for the drop shadow.

**dropShadowThickness**
Specifes the thickness of the drop shadow in units of pixels.

**verticalOffset**
Specifes the percentage of the logo to be positioned vertically off the main matte. By default the logo is centered horizontally and positioned vertically by this amount above the matte. This resource is ignored if y is specified.

**x** Specifes the *x* origin for the logo in units of pixels. This resource overrides the default horizontal centering of the logo.

**y** Specifes the *y* origin for the logo in units of pixels. This resource overrides the default vertical positioning of the logo.

The default welcome message on the authentication screen may also be replaced with a message of the user's choice. The following resources are available to control content and positioning of the welcome message. The resources should be prefaced with the string **Vuelogin*greeting*** when specified.

**Greeting Resource Set**

| **Name** | **Class** | **Type** | **Default** |
|---|---|---|---|
| alignment | Alignment | char | ALIGNMENT_CENTER |
| background | Background | Pixel | dynamic |
| foreground | Foreground | Pixel | dynamic |
| fontList | FontList | FontList | dynamic |
| labelString | String | String | Welcome to %LocalHost% |
| x | X | Position | dynamic |
| y | X | Position | dynamic |

**alignment**
Specifies the alignment of text in the welcome message. Possible values are ALIGNMENT_BEGINNING, ALIGNMENT_CENTER, and ALIGNMENT_END.

**background**
Specifes the background color for the welcome message.

**foreground**
Specifes the foreground color for the welcome message.

**fontList**
Specifes the font to use for the welcome message.

**labelString**
Specifes the text to use in the welcome message. Multiple lines can be specified by including new-line characters, "\n", in the text. If the token *%LocalHost%* is included in the text, it will be replaced with the name of the host providing login service.

**x**
Specifes the *x* origin for the welcome message in units of pixels. By default the welcome message is centered horizontally in the login matte. While in the matte it is clipped to the matte boundaries. If it is positioned outside the matte, it may extend to the screen boundaries.

**y**
Specifes the *y* origin for the welcome message in units of pixels. By default the message is positioned slightly above the login area of the login matte.

## XDMCP ACCESS CONTROL

The database file specified by the **Vuelogin.accessFile** resource provides information which *vuelogin* uses to control access from displays requesting XDMCP service. This file contains entries which control the response to Direct and Broadcast queries.

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

When checking access for a particular display host, each entry is scanned in turn and the first matching entry determines the response.

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

Here is an example Xaccess file:

A

```
#
# Xaccess - XDMCP access control file
#

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

If XDMCP access is granted, a temporary file may be created in the **authDir** directory which contains authorization information for the X-terminal. It is deleted when the session starts.

## SESSION STARTUP

Three files are provided to assist in session startup. They can be replaced by other mechanisms via *vuelogin* resources.

### The Xstartup File

This file is typically a shell script. It is run as "root" and should be very careful about security. This is the place to put commands that display the message of the day or do other system-level functions on behalf of the user. Various environment variables are set for the use of this script:

| | |
|---|---|
| DISPLAY | is set to the associated display name |
| HOME | is set to the home directory of the user |
| PATH | is set to the value of the **systemPath** resource |
| USER | is set to the user name |
| SHELL | is set to the value of the **systemShell** resource |

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

### The Xsession File

This script reads in the user's personal environment from $HOME*/.vueprofile* and then invokes the desired session manager. It is run with the permissions of the authorized user, and has several environment variables pre-set. See the **Environment** section for a list of the pre-set variables.

### The Xreset File

Symmetrical with *Xstartup*, this script is run after the user session has terminated. Run as root, it should probably contain commands that undo the effects of commands in *Xstartup*, such as unmounting directories from file servers. The collection of environment variables that were passed to *Xstartup* are also given to *Xreset*.

## TYPICAL USAGE

*Vuelogin* is designed to operate in a wide variety of environments. The following setup is a good place to start, but may not be "typical" in many environments.

First off, the *vuelogin* configuration file should be set up. A good thing to do is to make a directory (ex. */usr/vue/config*) that contains all of the relevant files. Here is a typical configuration file, which could be named *Xconfig* :

```
Vuelogin.errorLogFile:      /usr/vue/config/Xerrors
Vuelogin.pidFile:           /usr/vue/config/Xpid
Vuelogin.accessFile:        /usr/vue/config/Xaccess
Vuelogin.servers:           /usr/vue/config/Xservers

Vuelogin*resources:         /usr/vue/config/Xresources
Vuelogin*startup:           /usr/vue/config/Xstartup
Vuelogin*session:           /usr/vue/config/Xsession
Vuelogin*reset:             /usr/vue/config/Xreset
```

As you can see, this file simply contains references to other files. Note that some of the resources are specified with "*" separating the components. These resources can be made unique for each different display, by replacing the "*" with the display-name. See the **Resources** section for a complete discussion.

The first file */usr/vue/config/Xservers* contains the list of displays to manage. Most workstations have only one display, numbered 0, so the file looks like this:

```
:0 Local local /usr/bin/X11/X :0
```

This keeps */usr/bin/X11/X* running on this display and manage a continuous cycle of sessions.

The file */usr/vue/config/Xerrors* contains error messages from *vuelogin* and anything output to stderr by *Xstartup, Xsession or Xreset*. When you have trouble getting *vuelogin* working, check this file to see if *vuelogin* has any clues to the trouble. *Xerrors* can become quite large and should be trimmed periodically.

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

## SOME OTHER POSSIBILITIES

You can also use *vuelogin* to run a single session at a time by specifying the server on the command line:

```
vuelogin -server ":0 HP-TVRX local /usr/bin/X11/X :0"
```

If you have an X terminal that supports the XDMCP protocol, an entry for that terminal in *Xservers* is not required. If you have a file server and all X terminals support XDMCP, then *Xservers* would contain no entries.

Configurations may contain combinations of local servers, X terminals without XDMCP, and X terminals with XDCMP.

## COPYRIGHT

Copyright 1988, Massachusetts Institute of Technology.
See *X(1)* for a full statement of rights and permissions.

## AUTHOR

*Vuelogin* is based on the MIT client *XDM*, authored by Keith Packard. Additional modifications were developed by Hewlett Packard.

## ORIGIN

Hewlett-Packard Company.

## SEE ALSO

**connect(2), login(1), getty(1M), sh(1), stty(1), tset(1), X(1), xinit(1M), xrdb(1),** and **XDMCP.**

## NAME

x11start - start the X11 window system

## SYNOPSIS

**x11start** [*options*]

## DESCRIPTION

**NOTE:** Beginning with the next release of HP-UX, *x11start* and its components (*xinit, sys.x11start,* and *sys.Xdefaults*) will *not* be supported. *vuelogin* (an enhanced version of *xdm*) will perform all start-up tasks, regardless of whether or not HP-VUE is in use. See the *HP Visual Environment User's Guide* and the *vuelogin* man page for information on *vuelogin.*

A *x11start* is a shell script that provides a standarized method for starting up the X Window System server and selected X clients when the Visual User Environment (HP-VUE) is not used. Specifically, it performs the following start-up tasks:

- PATH environment variable set-up appropriate for the X environment
- X server start-up
- selected client(s) start-up from a specific client file
- general user resource loading from a specific resource file

### Components

*x11start* encompasses the following components:

- the */usr/bin/x11start* script
- the */usr/bin/X11/xinit* program
- the default client script, */usr/lib/X11/sys.x11start*
- the default resouce file, */usr/lib/X11/sys.Xdefaults*

*The x11start script*

- creates a command to load resources from the appropriate resource file using *xrdb* (see *xrdb(1)*) (This command is not executed immediately but rather is assigned to the (exported) environment variable, **doxrdb** which is normally executed via the client script after the server has been started up.)
- inserts */usr/bin/X11* ahead of */usr/bin* in the **PATH** environment variable and appends */usr/vue/bin* to the end
- execs *xinit* as follows (see *xinit(1)*):

  xinit *clientscript options*

  where

  | | |
  |---|---|
  | *clientscript* | is *$HOME/.x11start* or */usr/lib/X11/sys.x11start* |
  | *options* | are the *xinit* options exactly as specified on the *x11start* command line (see **Options** below). |

*The xinit program*

- starts the server with the server arguments specified in *options*
- runs the client script, passing it the client arguments specified in *options*
- waits for either the server or client script to terminate and then terminates whatever remains running

*The default client script, /usr/lib/X11/sys.x11start*

- executes the environment variable, **$doxrdb** which was setup previously by *x11start* to load the appropriate resource file to the server
- runs both *mwm* and *hpterm* with all client arguments specified in *options*
- sleeps indefinitely so that the X Window System will not be shut down by *xinit* when all client script processes have terminated.

*The default resource file, /usr/lib/X11/sys.Xdefaults*

- sets the default *Foreground, Background* and *BorderColor* resources
- contains sample resources for a number of common clients.

*Customized client and resource files*

can be created by copying the default files to *$HOME/.x11start* and *$HOME/.Xdefaults*, respectively, and then customizing them. Customized files that exist (with the appropriate permissions) will be used by the *x11start* components in place of the default files.

### Options

*x11start options* are the same as *xinit* options except that a client script cannot be specified.

Arguments preceeding a "--" (double dash), or all arguments if "--" is not specified, are passed on to the client script and can be accessed as positional parameters within that script.

Arguments following a "--" that begins with a slash (/) or a period (.) identify the server. If a server is not explicity specified, *xinit* determines the server from **$DISPLAY** if defined; otherwise from **$XSERVERRC**. If neither of these are defined, *$HOME/.xserverrc* is used if it exists and is executable; otherwise *X :0* is used.

A

Arguments following a "--" that don't begin with a slash or period are passed on to the appropriate server.

See *xinit(1)* for more details.

## EXAMPLES

The following examples illustrate how *x11start* can be used to control the server and/or interact with the client script.

**x11start**
This starts the default *xinit* server and executes the client script (either */usr/lib/X11/sys.x11start or $HOME/.x11start*) without passing arguments to either.

**x11start -bg black -fn 24x36**
This starts the default *xinit* server and executes the client script, passing it all of the arguments. If the default client script is used, *-bg black -fn 24x36* is be passed to both *mwm* and *hpterm* since both of their default command lines contain "$@". If the default client script is used, the actual *xinit* command executed is:

```
xinit /usr/lib/X11/sys.x11start -bg black -fn 24x36
```

**x11start -fg white -- :1**
This starts the default server on display 1 and executes the client script with the arguments, *-fg white*. If the customized client script is used, the *xinit* command line is:

```
xinit $HOME/.x11start -fg white -- :1
```

**x11start -- Xhp -a2 -t 5**
This starts the server, *Xhp* with the arguents, *-a2 -t 5* and then executes the client script without any arguments.

## ENVIRONMENT VARIABLES

**DISPLAY**
**PATH**
**XSERVERRC**
**doxrgb**

## FILES

*/usr/lib/X11/sys.Xdefaults*
*/usr/lib/X11/sys.x11start*
*$HOME/.Xdefaults*
*$HOME/.x11start*
*$HOME/.xserverrc*

## ORIGIN

HP

## SEE ALSO

X(1), xinit(1), hpterm(1), mwm(1), xrdb(1)

A

## NAME

xauth - X authority file utility

## SYNOPSIS

**xauth** [ **-f** *authfile* ] [ **-vqib** ] [ *command arg ...* ]

## DESCRIPTION

The *xauth* program is used to edit and display the authorization information used in connecting to the X server. This program is usually used to extract authorization records from one machine and merge them in on another (as is the case when using remote logins or granting access to other users). Commands (described below) may be entered interactively, on the *xauth* command line, or in scripts. Note that this program does **not** contact the X server.

## OPTIONS

The following options may be used with *xauth*. They may be given individually (e.g., *-q -i*) or may combined (e.g., *-qi*).

**-f** *authfile*
: This option specifies the name of the authority file to use. By default, *xauth* will use the file specified by the XAUTHORITY environment variable or *.Xauthority* in the user's home directory.

**-q**
: This option indicates that *xauth* should operate quietly and not print unsolicited status messages. This is the default if an *xauth* command is is given on the command line or if the standard output is not directed to a terminal.

**-v**
: This option indicates that *xauth* should operate verbosely and print status messages indicating the results of various operations (e.g., how many records have been read in or written out). This is the default if *xauth* is reading commands from its standard input and its standard output is directed to a terminal.

**-i**
: This option indicates that *xauth* should ignore any authority file locks. Normally, *xauth* will refuse to read or edit any authority files that have been locked by other programs (usually *xdm* or another *xauth*).

**-b**
: This option indicates that *xauth* should attempt to break any authority file locks before proceeding. Use this option only to clean up stale locks.

## COMMANDS

The following commands may be used to manipulate authority files:

**add** *displayname protocolname hexkey*
: An authorization entry for the indicated display using the given protocol and key data is added to the authorization file. The data is specified as an even-lengthed string of hexadecimal digits, each pair representing one octet. The first digit of each pair gives the most significant 4 bits of the octet, and the second digit of the pair gives the least significant 4 bits. For example, a 32 character hexkey would represent a 128-bit value. A protocol name consisting of just a single period is treated as an abbreviation for *MIT-MAGIC-COOKIE-1.*

**[n]extract** *filename displayname...*
: Authorization entries for each of the specified displays are written to the indicated file. If the *nextract* command is used, the entries are written in a numeric format suitable for non-binary transmission (such as secure electronic mail). The extracted entries can be read back in using the *merge* and *nmerge* commands. If the filename consists of just a single dash, the entries will be written to the standard output.

**[n]list** [*displayname...*]
: Authorization entries for each of the specified displays (or all if no displays are named) are printed on the standard output. If the *nlist* command is used, entries will be shown in the numeric format used by the *nextract* command; otherwise, they are shown in a textual format. Key data is always displayed in the hexadecimal format given in the description of the *add* command.

**[n]merge** *[filename...]*
Authorization entries are read from the specified files and are merged into the authorization database, superceding any matching existing entries. If the *nmerge* command is used, the numeric format given in the description of the *extract* command is used. If a filename consists of just a single dash, the standard input will be read if it hasn't been read before.

**remove** *displayname...*
Authorization entries matching the specified displays are removed from the authority file.

**source** *filename*
The specified file is treated as a script containing *xauth* commands to execute. Blank lines and lines beginning with a sharp sign (#) are ignored. A single dash may be used to indicate the standard input, if it hasn't already been read.

**info** Information describing the authorization file, whether or not any changes have been made, and from where *xauth* commands are being read is printed on the standard output.

**exit** If any modifications have been made, the authority file is written out (if allowed), and the program exits. An end of file is treated as an implicit *exit* command.

**quit** The program exits, ignoring any modifications. This may also be accomplished by pressing the interrupt character.

**help** [*string*]
A description of all commands that begin with the given string (or all commands if no string is given) is printed on the standard output.

**?** A short list of the valid commands is printed on the standard output.

## DISPLAY NAMES

Display names for the *add*, [*n*]*extract*, [*n*]*list*, [*n*]*merge*, and *remove* commands use the same format as the DISPLAY environment variable and the common *-display* command line argument. Display-specific information (such as the screen number) is unnecessary and will be ignored. Same-machine connections (such as local-host sockets, shared memory, and the Internet Protocol hostname *localhost*) are referred to as *hostname*/unix:*displaynumber* so that local entries for different machines may be stored in one authority file.

## EXAMPLE

The most common use for *xauth* is to extract the entry for the current display, copy it to another machine, and merge it into the user's authority file on the remote machine:

```
% xauth extract - $DISPLAY | rsh otherhost xauth merge -
```

## ENVIRONMENT

This *xauth* program uses the following environment variables:

**XAUTHORITY**
to get the name of the authority file to use if the *-f* option isn't used.

**HOME** to get the user's home directory if XAUTHORITY isn't defined.

## FILES

*$HOME/.Xauthority*
default authority file if XAUTHORITY isn't defined.

## BUGS

Users that have unsecure networks should take care to use encrypted file transfer mechanisms to copy authorization entries between machines. Similarly, the *MIT-MAGIC-COOKIE-1* protocol is not very useful in unsecure environments. Sites that are interested in additional security may need to use encrypted authorization mechanisms such as Kerberos.

Spaces are currently not allowed in the protocol name. Quoting could be added for the truly perverse.

## COPYRIGHT

## AUTHOR

Jim Fulton, MIT X Consortium

## NAME

xclock - analog / digital clock for X

## SYNOPSIS

**xclock** [-*toolkitoption* ...] [-help] [-analog] [-digital] [-chime] [-hd *color*] [-hl *color*] [-update *seconds*] [-padding *number*]

## DESCRIPTION

The *xclock* program displays the time in analog or digital form. The time is continuously updated at a frequency which may be specified by the user. This program is nothing more than a wrapper around the Athena Clock widget.

## OPTIONS

*Xclock* accepts all of the standard X Toolkit command line options along with the additional options listed below:

**-help** This option indicates that a brief summary of the allowed options should be printed on the standard error.

**-analog** This option indicates that a conventional 12 hour clock face with tick marks and hands should be used. This is the default.

**-digital** This option indicates that a 24 hour digital clock should be used.

**-chime** This option indicates that the clock should chime once on the half hour and twice on the hour.

**-hd** *color*
This option specifies the color of the hands on an analog clock. The default is *black*.

**-hl** *color* This option specifies the color of the edges of the hands on an analog clock, and is only useful on color displays. The default is *black*.

**-update** *seconds*
This option specifies the frequency in seconds at which *xclock* should update its display. If the clock is obscured and then exposed, it will be updated immediately. A value of less than 30 seconds will enable a second hand on an analog clock. The default is 60 seconds.

**-padding** *number*
This option specifies the width in pixels of the padding between the window border and clock text or picture. The default is 10 on a digital clock and 8 on an analog clock.

## X DEFAULTS

This program uses the *Athena Clock* widget. It understands all of the core resource names and classes as well as:

**width** (class **Width**)
Specifies the width of the clock. The default for analog clocks is 164 pixels; the default for digital clocks is whatever is needed to hold the clock when displayed in the chosen font.

**height** (class **Height**)
Specifies the height of the clock. The default for analog clocks is 164 pixels; the default for digital clocks is whatever is needed to hold the clock when displayed in the chosen font.

**update** (class **Interval**)
Specifies the frequency in seconds at which the time should be redisplayed.

**foreground** (class **Foreground**)
Specifies the color for the tic marks. The default depends on whether *reverseVideo* is specified. If *reverseVideo* is specified the default is *white*, otherwise the default is *black*.

**hands** (class **Foreground**)
Specifies the color of the insides of the clock's hands. The default depends on whether *reverseVideo* is specified. If *reverseVideo* is specified the default is *white*, otherwise the default is *black*.

**highlight** (class **Foreground**)
: Specifies the color used to highlight the clock's hands. The default is depends on whether *reverseVideo* is specified. If *reverseVideo* is specified the default is *white*, otherwise the default is *black.*

**analog** (class **Boolean**)
: Specifies whether or not an analog clock should be used instead of a digital one. The default is True.

**chime** (class **Boolean**)
: Specifies whether or not a bell should be rung on the hour and half hour.

A

**padding** (class **Margin**)
: Specifies the amount of internal padding in pixels to be used. The default is 8.

**font** (class **Font**)
: Specifies the font to be used for the digital clock. Note that variable width fonts currently will not always display correctly.

## WIDGETS

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

```
XClock  xclock
        Clock  clock
```

## ENVIRONMENT

DISPLAY
: the default host and display number.

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

## FILES

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

## SEE ALSO

X(1), xrdb(1), time(3C), Athena Clock widget

## BUGS

*Xclock* believes the system clock.

When in digital mode, the string should be centered automatically.

## COPYRIGHT

Copyright 1988, Massachusetts Institute of Technology.
See *X(1)* for a full statement of rights and permissions.

## AUTHORS

Tony Della Fera (MIT-Athena, DEC)
Dave Mankins (MIT-Athena, BBN)
Ed Moy (UC Berkeley)

## NAME

xcmsdb - Xlib Screen Color Characterization Data utility

## SYNOPSIS

**xcmsdb** [-option ...] [*filename*]

## DESCRIPTION

*xcmsdb* is used to load, query, or remove Screen Color Characterization Data stored in properties on the root window of the screen. Screen Color Characterization Data is an integral part of Xlib, necessary for proper conversion between device-independent and device-dependent color specifications. Xlib uses the XDCCC_LINEAR_RGB_MATRICES and XDCCC_LINEAR_RGB_CORRECTION properties to store color characterization data for color monitors. It uses XDCCC_GRAY_SCREENWWHITEPOINT and XDCCC_GRAY_CORRECTION properties for gray scale monitors. Because Xlib allows the addition of Screen Color Characterization Function Sets, added function sets may place their Screen Color Characterization Data on other properties. This utility is unaware of these other properties, therefore, you will need to use a similar utility provided with the function set, or use the *xprop* utility. The ASCII readable contents of *filename* (or the standard input if no input file is given) are appropriately transformed for storage in properties, provided the **-query** or **-remove** options are not specified.

## OPTIONS

*xcmsdb* program accepts the following options:

**-query** This option attempts to read the XDCCC properties off the screen's root window. If successful, it transforms the data into a more readable format, then sends the data to standard out.

**-remove** This option attempts to remove the XDCCC properties on the screen's root window.

**-color** This option sets the query and remove options to only check for the XDCCC_LINEAR_RGB_MATRICES and XDCCC_LINEAR_RGB_CORRECTION properties. If the **-color** option is not set then the query and remove options check for all the properties.

**-format 32 | 16 | 8**
Specifies the property format (32, 16, or 8 bits per entry) for the XDCCC_LINEAR_RGB_CORRECTION property. Precision of encoded floating point values increases with the increase in bits per entry. The default is 32 bits per entry.

## SEE ALSO

xprop(1), Xlib documentation

## ENVIRONMENT

**DISPLAY**
to figure out which display and screen to use.

## BUGS

Unknown

## COPYRIGHT

Copyright 1990, Tektronix Inc.

## AUTHOR

Chuck Adams, Tektronix Inc.

## NAME

xhost - server access control program for X

## SYNOPSIS

**xhost** [[+-]name ...]

## DESCRIPTION

A

The *xhost* program is used to add and delete host names or user names to the list allowed to make connections to the X server. In the case of hosts, this provides a rudimentary form of privacy control and security. It is only sufficient for a workstation (single user) environment, although it does limit the worst abuses. Environments which require more sophisticated measures should implement the user-based mechanism, or use the hooks in the protocol for passing other authentication data to the server.

Hostnames that are followed by two colons (::) are used in checking DECnet connections; all other hostnames are used for TCP/IP connections.

User names contain an at-sign (@). When Secure RPC is being used, the network independent netname (e.g., "unix.*uid@domainname*") can be specified, or a local user can be specified with just the username and a trailing at-sign (e.g., "joe@").

## OPTIONS

*Xhost* accepts the following command line options described below. For security, the options that effect access control may only be run from the "controlling host". For workstations, this is the same machine as the server. For X terminals, it is the login host.

[+]*name*
: The given *name* (the plus sign is optional) is added to the list allowed to connect to the X server. The name can be a host name or a user name.

-*name*
: The given *name* is removed from the list of allowed to connect to the server. The name can be a host name or a user name. Existing connections are not broken, but new connection attempts will be denied. Note that the current machine is allowed to be removed; however, further connections (including attempts to add it back) will not be permitted. Resetting the server (thereby breaking all connections) is the only way to allow local connections again.

+
: Access is granted to everyone, even if they aren't on the list (i.e., access control is turned off).

-
: Access is restricted to only those on the list (i.e., access control is turned on).

*nothing*
: If no command line arguments are given, a message indicating whether or not access control is currently enabled is printed, followed by the list of those allowed to connect. This is the only option that may be used from machines other than the controlling host.

## DIAGNOSTICS

For each name added to the access control list, a line of the form "*name* being added to access contro list" is printed. For each name removed from the access control list, a line of the form "*name* being removed from access contro list" is printed.

## FILES

/etc/X*.hosts

## SEE ALSO

X(1), Xserver(1)

## ENVIRONMENT

**DISPLAY**
: to get the default host and display to use.

## BUGS

You can't specify a display on the command line because **-display** is a valid command line argument (indicating that you want to remove the machine named *"display"* from the access list).

This is not really a bug, but the X server stores network addresses, not host names. If somehow you change a host's network address while the server is still running, *xhost* must be used to add the new address and/or remove the old address.

**COPYRIGHT**

Copyright 1988, Massachusetts Institute of Technology.
See *X(1)* for a full statement of rights and permissions.

**AUTHORS**

Bob Scheifler, MIT Laboratory for Computer Science,
Jim Gettys, MIT Project Athena (DEC).

## NAME

xinit - X Window System initializer

## SYNOPSIS

**xinit** [ [ *client* ] *options* ] [ -- [ *server* ] [ *display* ] *options* ]

## DESCRIPTION

The *xinit* program is used to start the X Window System server and a first client program on systems that cannot start X directly from */etc/init* or in environments that use multiple window systems. When this first client exits, *xinit* will kill the X server and then terminate.

A

On HP systems, the X Window System is normally started via the Visual User Environment (VUE) which uses *xdm* (X Display Manager) technology rather than *xinit* to start X. When VUE is not used, the normal method of starting X is via the *x11start* script which is simply a "wrapper" around *xinit* providing environment and command line setup appropriate for HP systems (see *x11start(1)*).

If no specific client program is given on the command line, *xinit* will look for a file in the user's home directory called *.xinitrc* to run as a shell script to start up client programs. If no such file exists, *xinit* will use the following as a default:

```
xterm -geometry +1+1 -n login
```

If no specific server program is given on the command line, *xinit* will look for a file in the user's home directory called *.xserverrc* to run as a shell script to start up the server. If no such file exists, *xinit* will use the following as a default:

```
X :0
```

Note that this assumes that there is a program named *X* in the current search path. However, servers are usually named *Xdisplaytype* where *displaytype* is the type of graphics display which is driven by this server. The site administrator should, therefore, make a link to the appropriate type of server on the machine, or create a shell script that runs *xinit* with the appropriate server.

An important point is that programs which are run by *.xinitrc* should be run in the background if they do not exit right away, so that they don't prevent other programs from starting up. However, the last long-lived program started (usually a window manager or terminal emulator) should be left in the foreground so that the script won't exit (which indicates that the user is done and that *xinit* should exit).

An alternate client and/or server may be specified on the command line. The desired client program and its arguments should be given as the first command line arguments to *xinit*. To specify a particular server command line, append a double dash (--) to the *xinit* command line (after any client and arguments) followed by the desired server command.

Both the client program name and the server program name must begin with a slash (/) or a period (.). Otherwise, they are treated as an arguments to be appended to their respective startup lines. This makes it possible to add arguments (for example, foreground and background colors) without having to retype the whole command line.

If an explicit server name is not given and the first argument following the double dash (--) is a colon followed by a digit, *xinit* will use that number as the display number instead of zero and will incorporate it into the $DISPLAY environment variable. All remaining arguments are appended to the server command line.

## EXAMPLES

Below are several examples of how command line arguments in *xinit* are used.

**xinit** This will start up a server named *X*, if *.xserverrc* doesn't exist, and run the user's *.xinitrc*, if it exists, or else start an *xterm*.

**xinit -- /usr/bin/X11/Xqdss :1**
This is how one could start a specific type of server on an alternate display.

**xinit -geometry =80x65+10+10 -fn 8x13 -j -fg white -bg navy**
This will start up a server named *X*, if *.xserverrc* doesn't exist, and will append the given arguments to the default *xterm* command. It will ignore *.xinitrc*.

**xinit -e widgets -- ./Xsun -l -c**
This will use the command *./Xsun -l -c* to start the server and will append the arguments *-e widgets* to the default *xterm* command.

**xinit remsh fasthost cpupig -display ws:1 -- :1 -a 2 -t 5**
This will start a server named *X* on display 1 with the arguments *-a 2 -t 5*. It will then start a remote shell on the machine **fasthost** in which it will run the command *cpupig*, telling it to display back on the local workstation.

A

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

```
xrdb -load $HOME/.Xresources
xsetroot -solid gray &
xclock -g 50x50-0+0 -bw 0 &
xload -g 50x50-50+0 -bw 0 &
xterm -g 80x24+0+0 &
xterm -g 80x24+0-0 &
mwm
```

Sites that want to create a common startup environment could simply create a default *.xinitrc* that references a site-wide startup file:

```
#!/bin/sh
. /usr/local/lib/site.xinitrc
```

Another approach is to write an X start-up script that runs *xinit* with a specific *client* script. Such X start-up scripts are usually named *x11*, *xstart*, or *startx* and are a convenient way to provide a simple interface for novice users. HP provides the X start-up script, */usr/bin/x11start* and the default *client* script, */usr/lib/X11/sys.x11start* for this purpose:

```
#!/bin/sh
 ...
xinit /usr/lib/X11/sys.x11start "$@"
```

where "$@" is simply additional *client* and/or *server* options that can be specified on the *x11start* command line. The default *client* script can be copied to the user's $HOME directory, customized and renamed *.x11start* in which case *x11start* will use this version in place of the default script (see *x11start(1)*).

## ENVIRONMENT VARIABLES

| | |
|---|---|
| **DISPLAY** | If not already set, this variable gets set to the name of the display to which clients should connect. |
| **XINITRC** | This variable specifies an init file containing shell commands to start up the initial windows. By default, *.xinitrc* in the home directory will be used. |
| **XSERVERRC** | This variable specifies an init file containing shell commands to start up the server. By default, *.xserverrc* in the home directory will be used. |

## FILES

| | |
|---|---|
| *.xinitrc* | default client script |
| *xterm* | client to run if *.xinitrc* does not exist |
| *.xserverrc* | default server script |
| *X* | server to run if *.xserverrc* does not exist |

A

## HP ENHANCEMENTS

HP has added a number of enhancements to the generic version of *xinit* provided by MIT. These enhancements may not be supported in future versions of *xinit.*

*DISPLAY* The generic version of *xinit* always resets the default value of $DISPLAY to *unix:0.0* regardless of its value prior to running *xinit.* This behavior has been changed so that the (exported) value of $DISPLAY outside of X will be maintained within X unless modified on the *xinit* command line. In addition, if $DISPLAY is not set outside of X, its default value will be initialized to *hostname*:0.0 where *hostname* is the name of the system invoking *xinit* as returned by *gethostname*(2).

*TTY Settings* The generic version of *xinit* always ran the *client* script in a new session (with a separate controlling terminal). Tty settings set outside of X, consequently, did not apply to X terminal windows. The *client* script is now run in the same session as *xinit* so that tty settings set prior to running *xinit* need not be reset within X.

## WARNINGS

*xinit,* as well as the *x11start* components, *x11start*, *sys.x11start* and *sys.Xdefaults* may not be supported in future releases. The *vuelogin* component of VUE will perform all of the tasks performed by these components and will become the supported method of starting up the X Window System both when VUE is and is not used.

## SEE ALSO

*X*(1), *x11start*(1), *Xserver*(1), *xterm*(1)

## COPYRIGHT

Copyright 1988, Massachusetts Institute of Technology.
See *X*(1) for a full statement of rights and permissions.

## AUTHOR

Bob Scheifler, MIT Laboratory for Computer Science

## NAME

xinitcolormap - initialize the X colormap

## SYNOPSIS

**xinitcolormap** [options]

## DESCRIPTION

This program is used to initialize the X colormap. Specific X colormap entries (pixel values) are made to correspond to specified colors. An initialized colormap is required by applications that assume a predefined colormap (e.g., many applications that use *Starbase* graphics).

*xinitcolormap* reads a colormap file to determine the allocation of colors in the X colormap. The name of the colormap file is determined by using (in the following order) the command line option [*-f colormapfile*], the *.Colormap* X default value or */usr/lib/X11/xcolormap*. If a colormap file is not found, then the following default colormap specification is assumed.

```
black (colormap entry 0)
white
red
yellow
green
cyan
blue
magenta (colormap entry 7)
```

*xinitcolormap* uses the *XStoreColor* and *XAllocColor libX11.a* calls to initialize the X colormap. The *xinitcolormap* program should be the first X client program run when the *X Window System* is started in order to assure that X colormap entries have the color associations specified in the colormap file. This could be done by running *xinitcolormap* as the first X client program in the *.x11start* file. Once *xinitcolormap* has been run, an X client program can use the initialized colors.

A colormap file is made up of lines of the form:

```
color
```

*color* is a one or two word color name (refer to the names in the file */usr/lib/X11/rgb.txt*), or optionally an initial sharp character followed by a numeric RGB specification (as used by the *libX.a* call *XParseColor*). The line number of a color specification in the colormap file determines the index of the color in the X colormap. Colors in the colormap file, for colormap entry 0 up to the last colormap entry to be initialized, must be specified. There should be no extra (blank or comment) lines in the colormap file. The first two entries in the colormap file must be black and white. Also, a color may be specified more than once in the colormap file.

## OPTIONS

**-f** *colormapfile*
: Specifies the file containing the colormap.

**-display** *display*
: Specifies the server to connect to; See *X(1)* for details.

**-c** *count*
: If *count* is specified then only the first *count* colors from the colormap file will be used in initializing the X colormap.

**-p**
: If the *-p* option is specified then the colormap file will be checked for proper syntax, but the X colormap will not be initialized.

**-k[ill]**
: If the *-k*[*ill*] option is specified, then the colormap entries allocated by a previous run of *xinitcolormap* will be deallocated and the colormap will not be re-initialized. All other options will be ignored except **-display** *display*.

## NOTES

*xinitcolormap* will only initialize the default colormap of the root window.

*xinitcolormap* assumes the first two colors specified are black and white.

*xinitcolormap* should not be run in the background. The X colormap is fully initialized only when *xinitcolormap* returns.

Running *xinitcolormap* a second time after X is started will deallocate those colors allocated by a previous run and attempt to allocate a new colormap using the new specifications. If other clients have allocated color cells that conflict with the new specifications, *xinitcolormap* will fail and the colormap will remain un-allocated.

A

The file */etc/newconfig/xcolormap* is a sample colormap file that corresponds to the *Starbase* default 256 entry colormap. The [*-c count*] option can be used to select a subset of the colors in this colormap file for initializing colormaps with up to 256 entries.

*xinitcolormap* uses XSetCloseDownMode with RetainPermanent to prevent the deallocation of the colormap. This means that *xinitcolormap* no longer spawns a daemon, and the only way for the user to be sure that *xinitcolormap* succeeded is to view the messages (or lack of) produced by *xinitcolormap*. If *x11start* is used, the output should be redirected from *xinitcolormap* to a log file.

*xinitcolormap* will not work on TrueColor visuals.

**FILES**

/usr/lib/X11/xcolormap
/usr/lib/X11/rgb.txt
/etc/newconfig/xcolormap
.x11start

## NAME

xload - system load average display for X

## SYNOPSIS

**xload** [*-toolkitoption* ...] [-scale *integer*] [-update *seconds*] [-hl *color*] [-highlight *color*] [-jumpscroll *pixels*] [-label *string*] [-nolabel] [-lights]

## DESCRIPTION

The *xload* program displays a periodically updating histogram of the system load average.

## OPTIONS

*Xload* accepts all of the standard X Toolkit command line options (see *X(1)*). The order of the options in unimportant. *xload* also accepts the following additional options:

A

**-hl** *color* or **-highlight** *color*
This option specifies the color of the scale lines.

**-jumpscroll** *number of pixels*
The number of pixels to shift the graph to the left when the graph reaches the right edge of the window. The default value is 1/2 the width of the current window. Smooth scrolling can be achieved by setting it to 1.

**-label** *string*
The string to put into the label above the load average.

**-nolabel** If this command line option is specified then no label will be displayed above the load graph.

**-lights** When specified, this option causes *xload* to display the current load average by using the keyboard leds; for a load average of *n*, xload lights the first *n* keyboard leds. This option turns off the usual screen display.

**-scale** *integer*
This option specifies the minimum number of tick marks in the histogram, where one division represents one load average point. If the load goes above this number, *xload* will create more divisions, but it will never use fewer than this number. The default is 1.

**-update** *seconds*
This option specifies the interval in seconds at which *xload* updates its display. The minimum amount of time allowed between updates is 1 second. The default is 10.

## RESOURCES

In addition to the resources available to each of the widgets used by *xload* there is one resource defined by the application itself.

**showLabel** (class **Boolean**)
If False then no label will be displayed.

## WIDGETS

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

```
XLoad  xload
        Paned  paned
                Label  label
                StripChart  load
```

## ENVIRONMENT

**DISPLAY**
to get the default host and display number.

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

**FILES**

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

**SEE ALSO**

X(1), xrdb(1), mem(4), Athena StripChart Widget.

**BUGS**

This program requires the ability to open and read the special system file */dev/kmem*. Sites that do not allow general access to this file should make *xload* belong to the same group as */dev/kmem* and turn on the *set group id* permission flag.

Reading /dev/kmem is inherently non-portable. Therefore, the routine used to read it (get_load.c) must be ported to each new operating system.

**COPYRIGHT**

**AUTHORS**

K. Shane Hartman (MIT-LCS) and Stuart A. Malone (MIT-LCS);
with features added by Jim Gettys (MIT-Athena), Bob Scheifler (MIT-LCS), Tony Della Fera (MIT-Athena), and Chris Peterson (MIT-LCS).

## NAME

xlsfonts - server font list displayer for X

## SYNOPSIS

**xlsfonts** [-options ...] [-fn pattern]

## DESCRIPTION

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

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

A

## OPTIONS

**-display** *host:dpy*
This option specifies the X server to contact.

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

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

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

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

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

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

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

**-o** This option indicates that *xlsfonts* should do an **OpenFont** (and **QueryFont**, if appropriate) rather than a **ListFonts**. This is useful if **ListFonts** or **ListFontsWithInfo** fail to list a known font (as is the case with some scaled font systems).

## SEE ALSO

X(1), Xserver(1), xset(1), xfd(1)

## ENVIRONMENT

**DISPLAY**
to get the default host and display to use.

## BUGS

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

## COPYRIGHT

Copyright 1988, Massachusetts Institute of Technology.
See *X(1)* for a full statement of rights and permissions.

## AUTHOR

Mark Lillibridge, MIT Project Athena; Jim Fulton, MIT X Consortium; Phil Karlton, SGI

**NAME**

xmodmap - utility for modifying keymaps in X

**SYNOPSIS**

**xmodmap** [-options ...] [filename]

**DESCRIPTION**

The *xmodmap* program is used to edit and display the keyboard *modifier map* and *keymap table* that are used by client applications to convert event keycodes into keysyms. It is usually run from the user's session startup script to configure the keyboard according to personal tastes.

**OPTIONS**

A

The following options may be used with *xmodmap*:

**-display** *display*
This option specifies the host and display to use.

**-help** This option indicates that a brief description of the command line arguments should be printed on the standard error channel. This will be done whenever an unhandled argument is given to *xmodmap.*

**-grammar**
This option indicates that a help message describing the expression grammar used in files and with -e expressions should be printed on the standard error.

**-verbose** This option indicates that *xmodmap* should print logging information as it parses its input.

**-quiet** This option turns off the verbose logging. This is the default.

**-n** This option indicates that *xmodmap* should not change the mappings, but should display what it would do, like *make(1)* does when given this option.

**-e** *expression*
This option specifies an expression to be executed. Any number of expressions may be specified from the command line.

**-pm** This option indicates that the current modifier map should be printed on the standard output.

**-pk** This option indicates that the current keymap table should be printed on the standard output.

**-pke** This option indicates that the current keymap table should be printed on the standard output in the form of expressions that can be fed back to *xmodmap*.

**-pp** This option indicates that the current pointer map should be printed on the standard output.

**-** A lone dash means that the standard input should be used as the input file.

The *filename* specifies a file containing *xmodmap* expressions to be executed. This file is usually kept in the user's home directory with a name like *.xmodmaprc.*

**EXPRESSION GRAMMAR**

The *xmodmap* program reads a list of expressions and parses them all before attempting to execute any of them. This makes it possible to refer to keysyms that are being redefined in a natural way without having to worry as much about name conflicts.

**keycode** *NUMBER = KEYSYMNAME ...*
The list of keysyms is assigned to the indicated keycode (which may be specified in decimal, hex or octal). The first keysym in the list is assigned to the keycode when no modifier key is used with the key, the second when the shift modifier key is used, the third when mod1 is used and the fourth when both shift and mod1 are used with the key (the standard translation makes use of only the first four keysyms in the list).

**keycode any** = *KEYSYMNAME ...*
If no existing key has the specified list of keysyms assigned to it, a spare key on the keyboard is selected and the keysyms are assigned to it. The list of keysyms may be

specified in decimal, hex or octal.

**keysym** *KEYSYMNAME = KEYSYMNAME ...*
: The *KEYSYMNAME* on the left hand side is translated into matching keycodes and used to perform the corresponding set of **keycode** expressions. The list of keysym names may be found in the header file *<X11/keysymdef.h>* (without the *XK_* prefix) or the keysym database */usr/lib/X11/XKeysymDB*. Note that if the same keysym is bound to multiple keys, the expression is executed for each matching keycode.

**clear** *MODIFIERNAME*
: This removes all entries in the modifier map for the given modifier, where valid name are: **Shift**, **Lock**, **Control**, **Mod1**, **Mod2**, **Mod3**, **Mod4**, and **Mod5** (case does not matter in modifier names, although it does matter for all other names). For example, "clear Lock" will remove all any keys that were bound to the shift lock modifier.

**add** *MODIFIERNAME = KEYSYMNAME ...*
: This adds all keys containing the given keysyms to the indicated modifier map. The keysym names are evaluated after all input expressions are read to make it easy to write expressions to swap keys (see the EXAMPLES section).

**remove** *MODIFIERNAME = KEYSYMNAME ...*
: This removes all keys containing the given keysyms from the indicated modifier map. Unlike **add**, the keysym names are evaluated as the line is read in. This allows you to remove keys from a modifier without having to worry about whether or not they have been reassigned.

**pointer = default**
: This sets the pointer map back to its default settings (button 1 generates a code of 1, button 2 generates a 2, etc.).

**pointer** = *NUMBER ...*
: This sets to pointer map to contain the indicated button codes. The list always starts with the first physical button.

Lines that begin with an exclamation point (!) are taken as comments.

If you want to change the binding of a modifier key, you must also remove it from the appropriate modifier map.

## EXAMPLES

Many pointers are designed such that the first button is pressed using the index finger of the right hand. People who are left-handed frequently find that it is more comfortable to reverse the button codes that get generated so that the primary button is pressed using the index finger of the left hand. This could be done on a 3 button pointer as follows:

```
% xmodmap -e "pointer = 3 2 1"
```

Many editor applications support the notion of Meta keys (similar to Control keys except that Meta is held down instead of Control). However, some servers do not have a Meta keysym in the default keymap table, so one needs to be added by hand. The following command will attach Meta to the Select key. It also takes advantage of the fact that applications that need a Meta key simply need to get the keycode and don't require the keysym to be in the first column of the keymap table. This means that applications that are looking for a Multi_key (including the default modifier map) won't notice any change.

```
% xmodmap -e "keysym Select = Select Meta_L"
```

One of the more simple, yet convenient, uses of *xmodmap* is to set the keyboard's "rubout" key to generate an alternate keysym. This frequently involves exchanging Backspace with Delete to be more comfortable to the user. If the *ttyModes* resource in *xterm* is set as well, all terminal emulator windows will use the same key for erasing characters:

```
% xmodmap -e "keysym BackSpace = Delete"
% echo "XTerm*ttyModes:  erase ^?" | xrdb -merge
```

Some keyboards do not automatically generate less than and greater than characters when the comma and period keys are shifted. This can be remedied with *xmodmap* by resetting the bindings for the comma and period with the following scripts:

```
!
! make shift-, be < and shift-. be >
!
keysym comma = comma less
keysym period = period greater
```

A

One of the more irritating differences between keyboards is the location of the Control and Shift Lock keys. A common use of *xmodmap* is to swap these two keys as follows:

```
!
! Swap Caps_Lock and Control_L
!
remove Lock = Caps_Lock
remove Control = Control_L
keysym Control_L = Caps_Lock
keysym Caps_Lock = Control_L
add Lock = Caps_Lock
add Control = Control_L
```

The *keycode* command is useful for assigning the same keysym to multiple keycodes. Although unportable, it also makes it possible to write scripts that can reset the keyboard to a known state. The following script sets the backspace key to generate Delete (as shown above), flushes all existing caps lock bindings, makes the CapsLock key be a control key, make F5 generate Escape, and makes Break/Reset be a shift lock.

```
!
! On the HP, the following keycodes have key caps as listed:
!
!   101 Backspace
!    55 Caps
!    14 Ctrl
!    15 Break/Reset
!    86 Stop
!    89 F5
!
keycode 101 = Delete
keycode 55 = Control_R
clear Lock
add Control = Control_R
keycode 89 = Escape
keycode 15 = Caps_Lock
add Lock = Caps_Lock
```

**ENVIRONMENT**

**DISPLAY**

to get default host and display number.

**SEE ALSO**

X(1), *Xlib* documentation on key and pointer events

**BUGS**

Every time a **keycode** expression is evaluated, the server generates a *MappingNotify* event on every client. This can cause some thrashing. All of the changes should be batched together and done at once. Clients that receive keyboard input and ignore *MappingNotify* events will not notice any changes made to keyboard mappings.

*Xmodmap* should generate "add" and "remove" expressions automatically whenever a keycode that is already bound to a modifier is changed.

There should be a way to have the *remove* expression accept keycodes as well as keysyms for those times when you really mess up your mappings.

**COPYRIGHT**

**AUTHOR**

Jim Fulton, MIT X Consortium, rewritten from an earlier version by David Rosenthal of Sun Microsystems.

## NAME

xpr - print an X window dump

## SYNOPSIS

**xpr** [ **-device** *devtype* ] [ **-scale** *scale* ] [ **-height** *inches* ] [ **-width** *inches* ] [ **-left** *inches* ] [ **-top** *inches* ] [ **-header** *string* ] [ **-trailer** *string* ] [ **-landscape** ] [ **-portrait** ] [ **-plane** *number* ] [ **-gray** ] [ **-rv** ] [ **-compact** ] [ **-output** *filename* ] [ **-append** *filename* ] [ **-noff** ] [ **-split** *n* ] [ **-psfig** ] [ **-density** *dpi* ] [ **-cutoff** *level* ] [ **-noposition** ] [ **-gamma** *correction* ] [ **-render** *algorithm* ] [ **-slide** ] [ *filename* ]

## DESCRIPTION

A

*xpr* takes as input a window dump file produced by *xwd(1)* and formats it for output on PostScript printers, the Digital LN03 or LA100, the IBM PP3812 page printer, the HP LaserJet (or other PCL printers), or the HP PaintJet. If no file argument is given, the standard input is used. By default, *xpr* prints the largest possible representation of the window on the output page. Options allow the user to add headers and trailers, specify margins, adjust the scale and orientation, and append multiple window dumps to a single output file. Output is to standard output unless **-output** is specified.

### Command Options

**-device** *devtype*

Specifies the device on which the file will be printed. Currently supported:

| | |
|---|---|
| **la100** | Digital LA100 |
| **ljet** | HP LaserJet series and other monochrome PCL devices such as ThinkJet, QuietJet, RuggedWriter, HP2560 series, and HP2930 series printers |
| **ln03** | Digital LN03 |
| **pjet** | HP PaintJet (color mode) |
| **pjetxl** | HP HP PaintJet XL Color Graphics Printer (color mode) |
| **pp** | IBM PP3812 |
| **ps** | PostScript printer |

The default is LaserJet. **-device lw** (LaserWriter) is equivalent to -device ps and is provided only for backwards compatibility.

**-scale** *scale*

Affects the size of the window on the page. The PostScript, LN03, and HP printers are able to translate each bit in a window pixel map into a grid of a specified size. For example each bit might translate into a 3x3 grid. This would be specified by **-scale** *3*. By default a window is printed with the largest scale that will fit onto the page for the specified orientation.

**-height** *inches*

Specifies the maximum height of the page.

**-width** *inches*

Specifies the maximum width of the page.

**-left** *inches*

Specifies the left margin in inches. Fractions are allowed. By default the window is centered in the page.

**-top** *inches*

Specifies the top margin for the picture in inches. Fractions are allowed.

**-header** *string*

Specifies a header string to be printed above the window.

**-trailer** *string*

Specifies a trailer string to be printed below the window.

**-landscape**

Forces the window to printed in landscape mode. By default a window is printed such that its longest side follows the long side of the paper.

**-plane** *number*
: Specifies which bit plane to use in an image. The default is to use the entire image and map values into black and white based on color intensities.

**-gray** *2 | 3 | 4*
: Uses a simple 2x2, 3x3, or 4x4 gray scale conversion on a color image, rather than mapping to strictly black and white. This doubles, triples, or quadruples the effective width and height of the image.

**-portrait**
: Forces the window to be printed in portrait mode. By default a window is printed such that its longest side follows the long side of the paper.

**-rv** Forces the window to be printed in reverse video.

**-compact**
: Uses simple run-length encoding for compact representation of windows with lots of white pixels.

**-output** *filename*
: Specifies an output file name. If this option is not specified, standard output is used.

**-append** *filename*
: Specifies a filename previously produced by *xpr* to which the window is to be appended.

**-noff** When specified in conjunction with **-append**, the window will appear on the same page as the previous window.

**-split** *n* This option allows the user to split a window onto several pages. This might be necessary for very large windows that would otherwise cause the printer to overload and print the page in an obscure manner.

**-psfig** Suppress translation of the PostScript picture to the center of the page.

**-density** *dpi*
: Indicates what dot-per-inch density should be used by the HP printer.

**-cutoff** *level*
: Changes the intensity level where colors are mapped to either black or white for monochrome output on a LaserJet printer. The *level* is expressed as percentage of full brightness. Fractions are allowed.

**-noposition**
: This option causes header, trailer, and image positioning command generation to be bypassed for LaserJet, PaintJet and PaintJet XL printers.

**-gamma** *correction*
: This changes the intensity of the colors printed by PaintJet XL printer. The *correction* is a floating point value in the range 0.00 to 3.00. Consult the operator's manual to determine the correct value for the specific printer.

**-render** *algorithm*
: This allows PaintJet XL printer to render the image with the best quality versus performance tradeoff. Consult the operator's manual to determine which *algorithms* are available.

**-slide** This option allows overhead transparencies to be printed using the PaintJet and PaintJet XL printers.

## SEE ALSO

xwd(1), xwud(1), X(1)

## LIMITATIONS

The current version of *xpr* can generally print out on the LN03 most X windows that are not larger than two-thirds of the screen. For example, it will be able to print out a large Emacs window, but it will usually fail when trying to print out the entire screen. The LN03 has memory limitations that can cause it to incorrectly print very large or complex windows. The two most common errors encountered are "band too complex" and "page memory exceeded." In the first case, a window may have a particular six pixel row that contains too many changes (from black to white to black). This will cause the printer to drop part of the line and possibly parts of the rest of the page. The printer will flash the number '1' on its front panel when this problem occurs. A possible solution to this problem is to increase the scale of the picture, or to split the picture onto two or more pages. The second problem, "page memory exceeded," will occur if the picture contains too much black, or if the picture contains complex half-tones such as the background color of a display. When this problem occurs the printer will automatically split the picture into two or more pages. It may flash the number '5' on its from panel. There is no easy solution to this problem. It will probably be necessary to either cut and paste, or to rework the application to produce a less complex picture.

There are several limitations on the LA100 support: the picture will always be printed in portrait mode, there is no scaling, and the aspect ratio will be slightly off.

Support for PostScript output currently cannot handle the **-append**, **-noff** or **-split** options.

The **-compact** option is *only* supported for PostScript output. It compresses white space but not black space, so it is not useful for reverse-video windows.

For color images, should map directly to PostScript image support.

## HP PRINTERS

If no **-density** is specified on the command line 300 dots per inch will be assumed for *ljet* and 90 dots per inch for *pjet*. Allowable *density* values for a LaserJet printer are 300, 150, 100, and 75 dots per inch. Consult the operator's manual to determine densities supported by other printers.

If no **-scale** is specified the image will be expanded to fit the printable page area.

The default printable page area is 8x10.5 inches. Other paper sizes can be accommodated using the **-height** and **-width** options.

Note that a 1024x768 image fits the default printable area when processed at 100 dpi with scale=1, the same image can also be printed using 300 dpi with scale=3 but will require considerably more data be transferred to the printer.

*xpr* may be tailored for use with monochrome PCL printers other than the LaserJet. To print on a ThinkJet (HP2225A) *xpr* could be invoked as:

```
xpr -density 96 -width 6.667 filename
```

or for black-and-white output to a PaintJet:

```
xpr -density 180 filename
```

The monochrome intensity of a pixel is computed as 0.30*R + 0.59*G + 0.11*B. If a pixel's computed intensity is less than the **-cutoff** level it will print as white. This maps light-on-dark display images to black-on-white hardcopy. The default cutoff intensity is 50% of full brightness. Example: specifying **-cutoff 87.5** moves the white/black intensity point to 87.5% of full brightness.

A LaserJet printer must be configured with sufficient memory to handle the image. For a full page at 300 dots per inch approximately 2MB of printer memory is required.

Color images are produced on the PaintJet at 90 dots per inch. The PaintJet is limited to sixteen colors from its 330 color palette on each horizontal print line. *xpr* will issue a warning message if more than sixteen colors are encountered on a line. *xpr* will program the PaintJet for the first sixteen colors encountered on each line and use the nearest matching programmed value for other colors present on the line.

Specifying the **-rv**, reverse video, option for the PaintJet will cause black and white to be interchanged on the output image. No other colors are changed.

Multiplane images must be recorded by *xwd* in *ZPixmap* format. Single plane (monochrome) images may be in either *XYPixmap* or *ZPixmap* format.

Some PCL printers do not recognize image positioning commands. Output for these printers will not be centered on the page and header and trailer strings may not appear where expected.

The **-gamma** and **-render** options are supported only on the PaintJet XL printers.

The **-slide** option is not supported for LaserJet printers.

The **-split** option is not supported for HP printers.

The **-gray** option is not supported for HP or IBM printers.

## COPYRIGHT

## AUTHORS

Michael R. Gretzinger, MIT Project Athena, Jose Capo, MIT Project Athena (PP3812 support), Marvin Solomon, University of Wisconsin, Bob Scheifler, MIT, Angela Bock and E. Mike Durbin, Rich Inc. (grayscale), Larry Rupp, HP (HP printer support).

**NAME**

xrdb - X server resource database utility

**SYNOPSIS**

**xrdb** [-option ...] [*filename*]

**DESCRIPTION**

A

*Xrdb* is used to get or set the contents of the RESOURCE_MANAGER property on the root window of screen 0, or the SCREEN_RESOURCES property on the root window of any or all screens, or everything combined. You would normally run this program from your X startup file. Most X clients use the RESOURCE_MANAGER and SCREEN_RESOURCES properties to get user preferences about color, fonts, and so on for applications. Having this information in the server (where it is available to all clients) instead of on disk, solves the problem in previous versions of X that required you to maintain *defaults* files on every machine that you might use. It also allows for dynamic changing of defaults without editing files. The RESOURCE_MANAGER property is used for resources that apply to all screens of the display. The SCREEN_RESOURCES property on each screen specifies additional (or overriding) resources to be used for that screen. (When there is only one screen, SCREEN_RESOURCES is normally not used, all resources are just placed in the RESOURCE_MANAGER property.) The file specified by *filename* (or the contents from standard input if - or no filename is given) is optionally passed through the C preprocessor with the following symbols defined, based on the capabilities of the server being used:

**BITS_PER_RGB=num**
the number of significant bits in an RGB color specification. This is the log base 2 of the number of distinct shades of each primary that the hardware can generate. Note that it usually is not related to PLANES.

**CLASS=visualclass**
one of StaticGray, GrayScale, StaticColor, PseudoColor, TrueColor, DirectColor. This is the visual class of the root window of the default screen.

**COLOR** defined only if CLASS is one of StaticColor, PseudoColor, TrueColor, or DirectColor.

**HEIGHT=num**
the height of the default screen in pixels.

**SERVERHOST=hostname**
the hostname portion of the display to which you are connected.

**HOST=hostname**
the same as SERVERHOST.

**CLIENTHOST=hostname**
the name of the host on which *xrdb* is running.

**PLANES=num**
the number of bit planes (the depth) of the root window of the default screen.

**RELEASE=num**
the vendor release number for the server. The interpretation of this number will vary depending on VENDOR.

**REVISION=num**
the X protocol minor version supported by this server (currently 0).

**VERSION=num**
the X protocol major version supported by this server (should always be 11).

**VENDOR=vendor**
a string specifying the vendor of the server.

**WIDTH=num**
the width of the default screen in pixels.

**X_RESOLUTION=num**
the x resolution of the default screen in pixels per meter.

**Y_RESOLUTION=num**
the y resolution of the default screen in pixels per meter. Lines that begin with an exclamation mark (!) are ignored and may be used as comments. Note that since *xrdb* can read from standard input, it can be used to the change the contents of properties directly from a terminal or from a shell script.

## OPTIONS

*xrdb* program accepts the following options:

**-help** This option (or any unsupported option) will cause a brief description of the allowable options and parameters to be printed.

A

**-display** *display*
This option specifies the X server to be used; see *X(1)*. It also specifies the screen to use for the *-screen* option, and it specifies the screen from which preprocessor symbols are derived for the *-global* option.

**-all** This option indicates that operation should be performed on the screen-independent resource property (RESOURCE_MANAGER), as well as the screen-specific property (SCREEN_RESOURCES) on every screen of the display. For example, when used in conjunction with *-query*, the contents of all properties are output. For *-load* and *-merge*, the input file is processed once for each screen. The resources which occur in common in the output for every screen are collected, and these are applied as the screen-independent resources. The remaining resources are applied for each individual per-screen property. This the default mode of operation.

**-global** This option indicates that the operation should only be performed on the screen-independent RESOURCE_MANAGER property.

**-screen** This option indicates that the operation should only be performed on the SCREEN_RESOURCES property of the default screen of the display.

**-screens** This option indicates that the operation should be performed on the SCREEN_RESOURCES property of each screen of the display. For *-load* and *-merge*, the input file is processed for each screen.

**-n** This option indicates that changes to the specified properties (when used with *-load* or *-merge*) or to the resource file (when used with *-edit*) should be shown on the standard output, but should not be performed.

**-quiet** This option indicates that warning about duplicate entries should not be displayed.

**-cpp** *filename*
This option specifies the pathname of the C preprocessor program to be used. Although *xrdb* was designed to use CPP, any program that acts as a filter and accepts the -D, -I, and -U options may be used.

**-nocpp** This option indicates that *xrdb* should not run the input file through a preprocessor before loading it into properties.

**-symbols**
This option indicates that the symbols that are defined for the preprocessor should be printed onto the standard output.

**-query** This option indicates that the current contents of the specified properties should be printed onto the standard output. Note that since preprocessor commands in the input resource file are part of the input file, not part of the property, they won't appear in the output from this option. The **-edit** option can be used to merge the contents of properties back into the input resource file without damaging preprocessor commands.

**-load** This option indicates that the input should be loaded as the new value of the specified properties, replacing whatever was there (i.e. the old contents are removed). This is the default action.

**-merge** This option indicates that the input should be merged with, instead of replacing, the current contents of the specified properties. Note that this option does a lexicographic sorted merge of the two inputs, which is almost certainly not what you want, but remains for backward compatibility.

**-remove** This option indicates that the specified properties should be removed from the server.

**-retain** This option indicates that the server should be instructed not to reset if *xrdb* is the first client. This never be necessary under normal conditions, since *xdm* and *xinit* always act as the first client.

**-edit** *filename*
This option indicates that the contents of the specified properties should be edited into the given file, replacing any values already listed there. This allows you to put changes that you have made to your defaults back into your resource file, preserving any comments or preprocessor lines.

A

**-backup** *string*
This option specifies a suffix to be appended to the filename used with **-edit** to generate a backup file.

**-D***name*[ =*value*]
This option is passed through to the preprocessor and is used to define symbols for use with conditionals such as *#ifdef.*

**-U***name* This option is passed through to the preprocessor and is used to remove any definitions of this symbol.

**-I***directory*
This option is passed through to the preprocessor and is used to specify a directory to search for files that are referenced with *#include.*

**FILES**

Generalizes *~/.Xdefaults* files.

**SEE ALSO**

X(1), Xlib Resource Manager documentation, Xt resource documentation

**ENVIRONMENT**

**DISPLAY**
to figure out which display to use.

**BUGS**

The default for no arguments should be to query, not to overwrite, so that it is consistent with other programs.

**COPYRIGHT**

Copyright 1991, Digital Equipment Corporation and MIT.

**AUTHORS**

Bob Scheifler, Phil Karlton, rewritten from the original by Jim Gettys

## NAME

xseethru - opens a transparent window into the image planes of a HP graphics display system while running X in the overlay planes.

## SYNOPSIS

**xseethru** [-geometry *geometry*] [-display *display*]

## DESCRIPTION

*xseethru* provides one method of viewing output from HP's high performance Starbase graphics applications while running X.

Normally, both X and Starbase utilize the image planes of the display and can not coexist together. However, if X is run in the overlay planes (by modifying /usr/lib/X11/X*screens), Starbase applications can be run via an X window and can utilize the normal Starbase output drivers that render to the image planes. When this is done, *xseethru* can be used to open a transparent window into the "underlaying" (and hidden) image planes to view a portion (or all, if the window is large enough) of the Starbase output.

Another method of viewing Starbase output while running X involves using HP's Starbase on X (SOX) subsystem. With this method, X is run as normal in the image planes and Starbase output is translated to X protocol so that it can be displayed and managed by X within a window in the standard X manner.

## OPTIONS

**-geometry** *geometry*
The transparent window is created with the specified size according to the geometry specification. See *X(1)* for details.

**-display** *display*
Specifies the sevrer to use; see *X(1)* for details.

## ENVIRONMENT

DISPLAY
To get the default host and display number.

## HARDWARE DEPENDENCIES

Only one display: 0, is supported on the HP 9000 Series 300.

*Xseethru* is only useful on the HP 98550A, HP 98720A, HP 98730A, HP 98735A/36A/36B, or HP 98705A/B/C Graphics Display Systems.

## ORIGIN

HP

## SEE ALSO

X(1)

**NAME**

X - a portable, network-transparent window system

**SYNOPSIS**

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

The X Consortium requests that the following names be used when referring to this software:

A

X
X Window System
X Version 11
X Window System, Version 11
X11

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

**DESCRIPTION**

X Window System servers run on computers with bitmap displays. The server distributes user input to and accepts output requests from various client programs through a variety of different interprocess communication channels. Although the most common case is for the client programs to be running on the same machine as the server, clients can be run transparently from other machines (including machines with different architectures and operating systems) as well.

X supports overlapping hierarchical subwindows and text and graphics operations, on both monochrome and color displays. For a full explanation of the functions that are available, see the *Xlib - C Language X Interface* manual, the *X Window System Protocol* specification, the *X Toolkit Intrinsics - C Language Interface* manual, and various toolkit documents.

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

Hewlett-Packard provides a graphical user environment called HP Visual User Environment (VUE). HP VUE is the user interface, enabling the user to control a workstation by directly manipulating graphic objects instead of typing commands on a command-line prompt. See the "HP Visual User Environment User's Guide" for complete information on HP VUE.

Hewlett-Packard does not provide or support the entire core MIT distribution. Many of these programs or clients are sample implementations, or perform tasks that are accomplished by other clients in Hewlett-Packard's Visual User Environment. The primary differences between the core MIT distribution and the Hewlett-Packard X11 release are listed below. See appendix A in "Using the X Window System" for a complete list of the clients supplied and supported with Hewlett-Packard's X11 release.

**Terminal Emulation**

*hpterm* is the primary terminal emulator. *xterm* is also provided and supported.

**Window Management**

*twm* is replaced by *mwm* and *vuewm*.

**Display Manager**

*xdm* is replaced by an enhanced version called *vuelogin*. terminal emulators.

**Bitmap Editing**

*bitmap* is replaced by *vueicon*.

**Font Display**

Handled by the terminal emulation option *-fn override*. *xfd* is supplied but not supported.

**Demos**

Obtained from the INTERWORKS users group.

A number of unsupported core MIT clients and miscellaneous utilities are provided in */usr/contrib/bin*. In addition, the entire core MIT distribution, compiled for Hewlett-Packard platforms, can be obtained from HP's users group INTERWORKS for a nominal fee. See the release notes for details.

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

A

## STARTING UP

Normally, the X Window System is started on Hewlett-Packard systems by *vuelogin*, which is an enhanced version of the MIT client *xdm*. *vuelogin* can be used to bring up a full VUE session, a light VUE session, or a fail-safe session that uses no other part of VUE. If *vuelogin* is not used, *xinit* may be used with *x11start*. See the man pages for these functions for more information.

## DISPLAY NAMES

From the user's perspective, every X server has a *display name* of the form:

*hostname:displaynumber.screennumber*

This information is used by the application to determine how it should connect to the server and which screen it should use by default (on displays with multiple monitors):

*hostname*

The *hostname* specifies the name of the machine to which the display is physically connected. If the hostname is not given, the most efficient way of communicating to a server on the same machine will be used.

*displaynumber*

The phrase "display" is usually used to refer to the collection of monitors that share a common keyboard and pointer (mouse, tablet, etc.). Most workstations tend to only have one keyboard, and therefore, only one display. Larger, multi-user systems, however, will frequently have several displays so that more than one person can be doing graphics work at once. To avoid confusion, each display on a machine is assigned a *display number* (beginning at 0) when the X server for that display is started. The display number must always be given in a display name.

*screennumber*

Some displays share a single keyboard and pointer among two or more monitors. Since each monitor has its own set of windows, each screen is assigned a *screen number* (beginning at 0) when the X server for that display is started. If the screen number is not given, then screen 0 will be used.

On POSIX systems, the default display name is stored in your DISPLAY environment variable. This variable is set automatically by the *xterm* terminal emulator. However, when you log into another machine on a network, you'll need to set DISPLAY by hand to point to your display. For example,

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

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

Finally, most X programs accept a command line option of **-display** *displayname* to temporarily override the contents of DISPLAY. This is most commonly used to pop windows on another person's screen or as part of a "remote shell" command to start an xterm pointing back to your display. For example,

```
% xload -display joesws:0 -geometry 100x100+0+0
% rsh big xterm -display myws:0 -ls </dev/null &
```

X servers listen for connections on a variety of different communications channels (network byte streams, shared memory, etc.). Since there can be more than one way of contacting a given server, The *hostname* part of the display name is used to determine the type of channel (also called a transport layer) to be used. X servers generally support the following types of connections:

*local*
: The hostname part of the display name should be the empty string. For example: *:0*, *:1*, and *:0.1*. The most efficient local transport is chosen.

A

*TCP/IP*
: The hostname part of the display name should be the server machine's IP address name. Full Internet names, abbreviated names, and IP addresses are all allowed. For example: *expo.lcs.mit.edu:0*, *expo:0*, *18.30.0.212:0*, *bigmachine:1*, and *hydra:0.1*.

## ACCESS CONTROL

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

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

*vuelogin/Xdm* initializes access control for the server, and also places authorization information in a file accessible to the user. Normally, the list of hosts from which connections are always accepted should be empty, so that only clients with are explicitly authorized can connect to the display. When you add entries to the host list (with *xhost*), the server no longer performs any authorization on connections from those machines. Be careful with this.

The file from which *Xlib* extracts authorization data can be specified with the environment variable **XAUTHORITY**, and defaults to the file **.Xauthority** in the home directory. *vuelogin/Xdm* uses **$HOME/.Xauthority** and will create it or merge in authorization records if it already exists when a user logs in.

If you use several machines, and share a common home directory across all of the machines by means of a network file system, then you never really have to worry about authorization files, the system should work correctly by default. Otherwise, as the authorization files are machine-independent, you can simply copy the files to share them. To manage authorization files, use *xauth*. This program allows you to extract records and insert them into other files. Using this, you can send authorization to remote machines when you login, if the remote machine does not share a common home directory with your local machine. Note that authorization information transmitted "in the clear" through a network file system or using *ftp* or *rcp* can be "stolen" by a network eavesdropper, and as such may enable unauthorized access. In many environments this level of security is not a concern, but if it is, you need to know the exact semantics of the particular authorization data to know if this is actually a problem.

## GEOMETRY SPECIFICATIONS

One of the advantages of using window systems instead of hardwired terminals is that applications don't have to be restricted to a particular size or location on the screen. Although the layout of windows on a display is controlled by the window manager that the user is running (described below), most X programs accept a command line argument of the form **-geometry** *WIDTHxHEIGHT+XOFF+YOFF* (where *WIDTH*, *HEIGHT*, *XOFF*, and *YOFF* are numbers) for specifying a preferred size and location for this application's main window.

The *WIDTH* and *HEIGHT* parts of the geometry specification are usually measured in either pixels or characters, depending on the application. The *XOFF* and *YOFF* parts are measured in pixels and are used to specify the distance of the window from the left or right and top and bottom edges of the screen, respectively. Both types of offsets are measured from the indicated edge of the screen to the corresponding edge of the window. The X offset may be specified in the following ways:

*+XOFF* The left edge of the window is to be placed *XOFF* pixels in from the left edge of the screen (i.e. the X coordinate of the window's origin will be *XOFF*). *XOFF* may be

negative, in which case the window's left edge will be off the screen.

*-XOFF* The right edge of the window is to be placed *XOFF* pixels in from the right edge of the screen. *XOFF* may be negative, in which case the window's right edge will be off the screen.

The Y offset has similar meanings:

*+YOFF* The top edge of the window is to be *YOFF* pixels below the top edge of the screen (i.e. the Y coordinate of the window's origin will be *YOFF*). *YOFF* may be negative, in which case the window's top edge will be off the screen.

*-YOFF* The bottom edge of the window is to be *YOFF* pixels above the bottom edge of the screen. *YOFF* may be negative, in which case the window's bottom edge will be off the screen.

Offsets must be given as pairs; in other words, in order to specify either *XOFF* or *YOFF* both must be present. Windows can be placed in the four corners of the screen using the following specifications:

| | |
|---|---|
| *+0+0* | upper left hand corner. |
| *-0+0* | upper right hand corner. |
| *-0-0* | lower right hand corner. |
| *+0-0* | lower left hand corner. |

In the following examples, a terminal emulator will be placed in roughly the center of the screen and a load average monitor, mailbox, and clock will be placed in the upper right hand corner:

```
xterm -fn 6x10 -geometry 80x24+30+200 &
xclock -geometry 48x48-0+0 &
xload -geometry 48x48-96+0 &
xbiff -geometry 48x48-48+0 &
```

## WINDOW MANAGERS

The layout of windows on the screen is controlled by special programs called *window managers*. Although many window managers will honor geometry specifications as given, others may choose to ignore them (requiring the user to explicitly draw the window's region on the screen with the pointer, for example).

Since window managers are regular (albeit complex) client programs, a variety of different user interfaces can be built. The Hewlett-Packard distribution comes with window managers named *mwm* and *vuewm* which support overlapping windows, popup menus, point-and-click or click-to-type input models, title bars, nice icons (and an icon manager for those who don't like separate icon windows).

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

## FONT NAMES

Collections of characters for displaying text and symbols in X are known as *fonts*. A font typically contains images that share a common appearance and look nice together (for example, a single size, boldness, slant, and character set). Similarly, collections of fonts that are based on a common type face (the variations are usually called roman, bold, italic, bold italic, oblique, and bold oblique) are called *families*.

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

An X server can obtain fonts from individual files stored in directories in the file system, or from one or more font servers, or from a mixtures of directories and font servers. The list of places the server looks when trying to find a font is controlled by its *font path*. Although most installations will choose to have the server start up with all of the commonly used font directories in the font path, the font path can be changed at any time with the *xset* program. However, it is important to remember that the directory names are on the server's machine, not on the application's. Usually, fonts usex by X servers and font servers can be found in subdirectories under */usr/lib/X11/fonts.*:

*/usr/lib/X11/fonts/iso_8859.1/75dpi*
: This directory contains bitmap fonts contributed by Adobe Systems, Inc., Digital Equipment Corporation, Bitstream, Inc., Bigelow and Holmes, and Sun Microsystems, Inc. for 75 dots per inch displays. An integrated selection of sizes, styles, and weights are provided for each family.

*/usr/lib/X11/fonts/iso_8859.1/100dpi*
: This directory contains 100 dots per inch versions of some of the fonts in the *75dpi* directory.

A

Bitmap font files are usually created by compiling a textual font description into binary form, using *bdftopcf*. Font databases are created by running the *mkfontdir* program in the directory containing the source or compiled versions of the fonts. Whenever fonts are added to a directory, *mkfontdir* should be rerun so that the server can find the new fonts. To make the server reread the font database, reset the font path with the *xset* program. For example, to add a font to a private directory, the following commands could be used:

```
% cp newfont.pcf ~/myfonts
% mkfontdir ~/myfonts
% xset fp rehash
```

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

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

might be abbreviated as:

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

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

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

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

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

To convert one of the resulting names into a font at a specific size, replace one of the first two zeros with a nonzero value. The field containing the first zero is for the pixel size; replace it with a specific height in pixels to name a font at that size. Alternatively, the field containing the second zero is for the point size; replace it with a specific size in decipoints (there are 722.7 decipoints to the inch) to name a font at that size. The last zero is an average width field, measured in tenths of pixels; some servers will anamorphically scale if this value is specified. See chapter 6 of *Using the X Window System*. to

## FONT SERVER NAMES

One of the following forms can be used to name a font server that accepts TCP connections:

tcp/*hostname:port*
tcp/*hostname:port/cataloguelist*

The *hostname* specifies the name (or decimal numeric address) of the machine on which the font server is running. The *port* is the decimal TCP port on which the font server is listening for connections. The *cataloguelist* specifies a list of catalogue names, with '+' as a separator.

Examples: *tcp/expo.lcs.mit.edu:7000*, *tcp/18.30.0.212:7001/all*.

**COLOR NAMES**

Most applications provide ways of tailoring (usually through resources or command line arguments) the colors of various elements in the text and graphics they display. A color can be specified either by an abstract color name, or by a numerical color specification. The numerical specification can identify a color in either device-dependent (RGB) or device-independent terms. Color strings are case-insensitive.

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

A

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

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

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

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

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

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

The eight primary colors can be represented as:

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

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

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

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

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

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

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

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

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

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

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

## KEYBOARDS

The X keyboard model is broken into two layers: server-specific codes (called *keycodes*) which represent the physical keys, and server-independent symbols (called *keysyms*) which represent the letters or words that appear on the keys. Two tables are kept in the server for converting keycodes to keysyms:

*modifier list*

Some keys (such as Shift, Control, and Caps Lock) are known as *modifier* and are used to select different symbols that are attached to a single key (such as Shift-a generates a capital A, and Control-l generates a control character ^L). The server keeps a list of keycodes corresponding to the various modifier keys. Whenever a key is pressed or released, the server generates an *event* that contains the keycode of the indicated key as well as a mask that specifies which of the modifier keys are currently pressed. Most servers set up this list to initially contain the various shift, control, and shift lock keys on the keyboard.

*keymap table*

Applications translate event keycodes and modifier masks into keysyms using a *keysym table* which contains one row for each keycode and one column for various modifier states. This table is initialized by the server to correspond to normal typewriter conventions. The exact semantics of how the table is interpreted to produce keysyms depends on the particular program, libraries, and language input method used, but the following conventions for the first four keysyms in each row are generally adhered to:

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

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

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

## OPTIONS

Most X programs attempt to use the same names for command line options and arguments. All applications written with the X Toolkit Intrinsics automatically accept the following options:

**-display** *display*
: This option specifies the name of the X server to use.

**-geometry** *geometry*
: This option specifies the initial size and location of the window.

**-bg** *color*, **-background** *color*
: Either option specifies the color to use for the window background.

**-bd** *color*, **-bordercolor** *color*
: Either option specifies the color to use for the window border.

**-bw** *number*, **-borderwidth** *number*
: Either option specifies the width in pixels of the window border.

A

**-fg** *color*, **-foreground** *color*
: Either option specifies the color to use for text or graphics.

**-fn** *font*, **-font** *font*
: Either option specifies the font to use for displaying text.

**-iconic**
: This option indicates that the user would prefer that the application's windows initially not be visible as if the windows had be immediately iconified by the user. Window managers may choose not to honor the application's request.

**-name**
: This option specifies the name under which resources for the application should be found. This option is useful in shell aliases to distinguish between invocations of an application, without resorting to creating links to alter the executable file name.

**-rv**, **-reverse**
: Either option indicates that the program should simulate reverse video if possible, often by swapping the foreground and background colors. Not all programs honor this or implement it correctly. It is usually only used on monochrome displays.

**+rv**
: This option indicates that the program should not simulate reverse video. This is used to override any defaults since reverse video doesn't always work properly.

**-selectionTimeout**
: This option specifies the timeout in milliseconds within which two communicating applications must respond to one another for a selection request.

**-synchronous**
: This option indicates that requests to the X server should be sent synchronously, instead of asynchronously. Since *Xlib* normally buffers requests to the server, errors do not necessarily get reported immediately after they occur. This option turns off the buffering so that the application can be debugged. It should never be used with a working program.

**-title** *string*
: This option specifies the title to be used for this window. This information is sometimes used by a window manager to provide some sort of header identifying the window.

**-xnllanguage** *language*[*_territory*][*.codeset*]
: This option specifies the language, territory, and codeset for use in resolving resource and other filenames.

**-xrm** *resourcestring*
: This option specifies a resource name and value to override any defaults. It is also very useful for setting resources that don't have explicit command line arguments.

## RESOURCES

To make the tailoring of applications to personal preferences easier, X provides a mechanism for storing default values for program resources (e.g. background color, window title, etc.) Resources are specified as strings that are read in from various places when an application is run. Program components are named in a hierarchical fashion, with each node in the hierarchy identified by a class and an instance name. At the top level is the class and instance name of the application itself. By convention, the class name of the application is the same as the program name, but with the first letter capitalized, although some programs that begin with the letter "x" also capitalize the second letter for historical reasons.

The precise syntax for resources is:

```
ResourceLine        =  Comment | IncludeFile | ResourceSpec | <empty line>
Comment             =  "!" {<any character except null or newline>}
IncludeFile         =  "#" WhiteSpace "include" WhiteSpace FileName WhiteSpace
FileName            =  <valid filename for operating system>
ResourceSpec        =  WhiteSpace ResourceName WhiteSpace ":" WhiteSpace Value
ResourceName        =  [Binding] {Component Binding} ComponentName
Binding             =  "." | "*"
WhiteSpace          =  {<space> | <horizontal tab>}
Component           =  "?" | ComponentName
ComponentName       =  NameChar {NameChar}
NameChar            =  "a"-"z" | "A"-"Z" | "0"-"9" | "_" | "-"
Value               =  {<any character except null or unescaped newline>}
```

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

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

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

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

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

When an application looks for the value of a resource, it specifies a complete path in the hierarchy, with both class and instance names. However, resource values are usually given with only partially specified names and classes, using pattern matching constructs. An asterisk (*) is a loose binding and is used to represent any number of intervening components, including none. A period (.) is a tight binding and is used to separate immediately adjacent components. A question mark (?) is used to match any single component name or class. A database entry cannot end in a loose binding; the final component (which cannot be "?") must be specified. The lookup algorithm searches the resource database for the entry that most closely matches (is most specific for) the full name and class being queried. When more than one database entry matches the full name and class, precedence rules are used to select just one. The full name and class are scanned from left to right (from highest level in the hierarchy to lowest), one component at a time. At each level, the corresponding component and/or binding of each matching entry is determined, and these matching components and bindings are compared according to precedence rules. Each of the rules is applied at each level, before moving to the next level, until a rule selects a single entry over all others. The rules (in order of precedence) are:

1. An entry that contains a matching component (whether name, class, or "?") takes precedence over entries that elide the level (that is, entries that match the level in a loose binding).
2. An entry with a matching name takes precedence over both entries with a matching class and entries that match using "?". An entry with a matching class takes precedence over entries that match using "?".
3. An entry preceded by a tight binding takes precedence over entries preceded by a loose binding.

Programs based on the X Tookit Intrinsics obtain resources from the following sources (other programs usually support some subset of these sources):

**RESOURCE_MANAGER root window property**
Any global resources that should be available to clients on all machines should be stored in the RESOURCE_MANAGER property on the root window of the first screen using the *xrdb* program. This is frequently taken care of when the user starts X through the display manager.

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

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

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

**-xrm** *resourcestring*
Resources can also be specified from the command line. The *resourcestring* is a single resource name and value as shown above. Note that if the string contains characters interpreted by the shell (e.g., asterisk), they must be quoted. Any number of **-xrm** arguments may be given on the command line.

Program resources are organized into groups called *classes*, so that collections of individual resources (each of which are called *instances*) can be set all at once. By convention, the instance name of a resource begins with a lowercase letter and class name with an upper case letter. Multiple word resources are concatenated with the first letter of the succeeding words capitalized. Applications written with the X Toolkit Intrinsics will have at least the following resources:

**background** (class **Background)**
This resource specifies the color to use for the window background.

**borderWidth** (class **BorderWidth)**
This resource specifies the width in pixels of the window border.

**borderColor** (class **BorderColor)**
This resource specifies the color to use.for the window border.

Most applications using the X Toolkit Intrinsics also have the resource **foreground** (class **Foreground**), specifying the color to use for text and graphics within the window.

By combining class and instance specifications, application preferences can be set quickly and easily. Users of color displays will frequently want to set Background and Foreground classes to particular defaults. Specific color instances such as text cursors can then be overridden without having to define all of the related resources. For example,

```
vueicon*Dashed: off
XTerm*cursorColor: gold
XTerm*multiScroll: on
XTerm*jumpScroll: on
XTerm*reverseWrap: on
XTerm*curses: on
XTerm*Font: 6x10
XTerm*scrollBar: on
XTerm*scrollbar*thickness: 5
XTerm*multiClickTime: 500
XTerm*charClass: 33:48,37:48,45-47:48,64:48
XTerm*cutNewline: off
XTerm*cutToBeginningOfLine: off
XTerm*titeInhibit: on
XTerm*ttyModes: intr ^c erase ^? kill ^u
XLoad*Background: gold
XLoad*Foreground: red
XLoad*highlight: black
XLoad*borderWidth: 0
hpterm*Geometry: 80x65-0-0
hpterm*Background: rgb:5b/76/86
hpterm*Foreground: white
hpterm*Cursor: white
hpterm*BorderColor: white
hpterm*Font: 6x10
```

If these resources were stored in a file called *.Xdefaults* in your home directory, they could be added to any existing resources in the server with the following command:

```
% xrdb -merge $HOME/.Xdefaults
```

This is frequently how user-friendly startup scripts merge user-specific defaults into any site-wide defaults. All sites are encouraged to set up convenient ways of automatically loading resources. See the *Xlib* manual section *Resource Manager Functions* for more information.

## EXAMPLES

The following is a collection of sample command lines for some of the more frequently used commands. For more information on a particular command, please refer to that command's manual page.

```
% xrdb $HOME/.Xdefaults
% xmodmap -e "keysym BackSpace = Delete"
% mkfontdir /usr/local/lib/X11/otherfonts
% xset fp+ /usr/local/lib/X11/otherfonts
% xmodmap $HOME/.keymap.km
% xsetroot -solid 'rgbi:.8/.8/.8'
% xset b 100 400 c 50 s 1800 r on
% xset q
% mwm
% xclock -geometry 48x48-0+0 -bg blue -fg white
% xlsfonts '*helvetica*'
% xwininfo -root
% xhost -joesworkstation
% xwd | xwud
% xterm -geometry 80x66-0-0 -name myxterm $*
```

## DIAGNOSTICS

A wide variety of error messages are generated from various programs. The default error handler in *Xlib* (also used by many toolkits) uses standard resources to construct diagnostic messages when errors occur. The defaults for these messages are usually stored in */usr/lib/X11/XErrorDB*. If this file is not present, error messages will be rather terse and cryptic.

When the X Toolkit Intrinsics encounter errors converting resource strings to the appropriate internal format, no error messages are usually printed. This is convenient when it is desirable to have one set of resources across a variety of displays (e.g. color vs. monochrome, lots of fonts vs. very few, etc.), although it can pose problems for trying to determine why an application might be failing. This behavior can be overridden by the setting the *StringConversionsWarning* resource.

A

To force the X Toolkit Intrinsics to always print string conversion error messages, the following resource should be placed in the *.Xdefaults* file in the user's home directory. This file is then loaded into the RESOURCE_MANAGER property using the *xrdb* program.

```
*StringConversionWarnings: on
```

To have conversion messages printed for just a particular application, the appropriate instance name can be placed before the asterisk:

```
xterm*StringConversionWarnings: on
```

## SEE ALSO

bdftopcf(1), bitmap(1), fs(1), hpterm(1) mkfontdir(1), mwm(1), xauth(1), xclock(1), xcmsdb(1), xfd(1), xhost(1), xinitcolor(1), xload(1), xlsfonts(1), xmodmap(1), xpr(1), xprop(1), xrdb(1), xrefresh(1), xset(1), xsetroot(1), xterm(1), xwd(1), xwininfo(1), xwud(1), Xserver(1), *Xlib - C Language X Interface*, and *X Toolkit Intrinsics - C Language Interface*

## COPYRIGHT

The following copyright and permission notice outlines the rights and restrictions covering most parts of the core distribution of the X Window System from MIT. Other parts have additional or different copyrights and permissions; see the individual source files.

## TRADEMARKS

X Window System is a trademark of MIT.

## AUTHORS

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

**NAME**

X - X Window System server

**SYNOPSIS**

**X** *:displaynumber* [-option] *ttyname*

**DESCRIPTION**

*X* is the generic name for the window system server. It is started by the *vuelogin(1X)* program which is typically run by *init(1M)*. Alternatively it may be started from the *xinit(1)* program, which is called by *x11start*. The *displaynumber* argument is used by clients in their DISPLAY environment variables to indicate which server to contact (large machines may have several displays attached). This number can be any number. If no number is specified 0 is used. This number is also used in determining the names of various startup files. The *ttyname* argument is passed in by *init* and isn't used.

A

The Hewlett-Packard server has support for the following protocols:

**TCP/IP**
The server listens on port 6000+N, where N is the display number.

**Local Socket IPC Mechanism**
The file name for the socket is "/usr/spool/sockets/X11/*" where "*" is the display number.

**Shared Memory IPC**
This is the default connection that the X Library will use to connect to an X server on the same machine if the DISPLAY environment variable is set to "local:*" or ":*" where "*" is the number of the display.

When the server starts up, it takes over the display. If you are running on a workstation whose console is the display, you cannot log into the console while the server is running.

**OPTIONS**

The following options can be given on the command line to the X server.

**-a** *number*
sets pointer acceleration (i.e. the ratio of how much is reported to how much the user actually moved the pointer).

**-auth** *authorization-file*
Specifies a file which contains a collection of authorization records used to authenticate access.

**bc** disables certain kinds of error checking, for bug compatibility with previous releases (e.g., to work around bugs in R2 and R3 xterms and toolkits). Deprecated.

**-bs** disables backing store support on all screens.

**-c** turns off key-click.

**c** *volume*
sets key-click volume (allowable range: 0-100).

**-co** *filename*
sets name of RGB color database.

**-dpi** *resolution*
sets the resolution of the screen, in dots per inch. To be used when the server cannot determine the screen size from the hardware.

**-f** *volume*
sets feep (bell) volume (allowable range: 0-100).

**-fc** *cursorFont*
sets default cursor font.

**-fn** *font* sets the default font.

**-fp** *fontPath*
sets the search path for fonts. This path is a comma separated list of directories which

the server searches for font databases.

**-help** prints a usage message.

**-I** causes all remaining command line arguments to be ignored.

**-logo** turns on the X Window System logo display in the screen-saver. There is currently no way to change this from a client.

**nologo** turns off the X Window System logo display in the screen-saver. There is currently no way to change this from a client.

**-p** *minutes*
sets screen-saver pattern cycle time in minutes.

**-pn** allows X server to run even if one or more communications mechanisms fails to initialize.

**-r** turns off keyboard auto-repeat.

**r** turns on keyboard auto-repeat.

**-s** *minutes*
sets screen-saver timeout time in minutes.

**-su** disables save under support on all screens.

**-t** *number*
sets pointer acceleration threshold in pixels (i.e. after how many pixels pointer acceleration should take effect).

**-to** *seconds*
sets default connection timeout in seconds.

**tty***xx* ignored, for servers started the ancient way (from init).

**-terminage**
causes server to terminate when all clients disconnect.

**v** sets video-on screen-saver preference. A window that changes regularly will be used to save the screen.

**-v** sets video-off screen-saver preference. The screen will be blanked to save the screen.

**-wm** forces the default backing-store of all windows to be WhenMapped; a less expensive way of getting backing-store to apply to all windows.

You can also have the X server connect to xdm(1) or vuelogin(1X) using
XDMCP. Although this is not typically useful as it doesn't allow xdm to manage the server process, it can be used to debug XDMCP implementations, and serves as a sample implementation of the server side of XDMCP. The following options control the behavior of XDMCP.

**-query** *host-name*
Enable XDMCP and send Query packets to the specified host.

**-broadcast**
Enable XDMCP and broadcast BroadcastQuery packets to the network. The first responding display manager will be chosen for the session.

**-indirect** *host-name*
Enable XDMCP and send IndirectQuery packets to the specified host.

**-port** *port-num*
Use an alternate port number for XDMCP packets. Must be specified before any -query, -broadcast or -indirect options. Default port number is 177.

**-once** Normally, the server keeps starting sessions, one after the other. This option makes the server exit after the first session is over.

**-class** *display-class*
XDMCP has an additional display qualifier used in resource lookup for display-specific

options. This option sets that value, by default it is "MIT-Unspecified" (not a very useful value).

**-cookie** *xdm-auth-bits*
When testing XDM-AUTHENTICATION-1, a private key is shared between the server and the manager. This option sets the value of that private data (not that it's very private, being on the command line and all...).

**-displayID** *display-id*
Yet another XDMCP specific value, this one allows the display manager to identify each display so that it can locate the shared key.

A

## RUNNING FROM INIT

Though X will usually be run by *vuelogin* from *init*, it is possible to run X directly from *init*. For information about running X from *vuelogin*, see the *vuelogin* man page.

To run X directly from *init*, it is necessary to modify */etc/inittab* and */etc/gettydefs*. Detailed information on these files may be obtained from the *inittab(4)* and *gettydefs(4)* man pages.

To run X from *init* on display 0, with a login *xterm* running on */dev/ttypf*, in init state 3, the following line must be added to */etc/inittab*:

```
X0:3:respawn:env PATH=/bin:/usr/bin/X11:/usr/bin  xinit -L ttyqf -- :0
```

To run X with a login *hpterm*, the following should be used instead:

```
X0:3:respawn:env PATH=/bin:/usr/bin/X11:/usr/bin  xinit hpterm =+1+1 -n login -L ttyqf --
:0
```

In addition, the following line must be added to */etc/gettydefs* (this should be a single line):

```
Xwindow# B9600 HUPCL PARENB CS7 # B9600 SANE PARENB CS7 ISTRIP IXANY
TAB3   #X login: #Xwindow
```

There should not be a getty running against the display whenever X is run from *xinit*.

## SECURITY

The sample server implements a simplistic authorization protocol, MIT-MAGIC-COOKIE-1 which uses data private to authorized clients and the server. This is a rather trivial scheme; if the client passes authorization data which is the same as the server has, it is allowed access. This scheme is inferior to host-based access control mechanisms in environments with unsecure networks as it allows any host to connect, given that it has discovered the private key. But in many environments, this level of security is better than the host-based scheme as it allows access control per-user instead of per-host.

In addition, the server provides support for a DES-based authorization scheme, XDM-AUTHORIZATION-1, which is more secure (given a secure key distribution mechanism), but as DES is not generally distributable, the implementation is missing routines to encrypt and decrypt the authorization data. This authorization scheme can be used in conjunction with XDMCP's authentication scheme, XDM-AUTHENTICATION-1 or in isolation.

The authorization data is passed to the server in a private file named with the **-auth** command line option. Each time the server is about to accept the first connection after a reset (or when the server is starting), it reads this file. If this file contains any authorization records, the local host is not automatically allowed access to the server, and only clients which send one of the authorization records contained in the file in the connection setup information will be allowed access. See the *Xau* manual page for a description of the binary format of this file. Maintenance of this file, and distribution of its contents to remote sites for use there is left as an exercise for the reader.

The sample server also uses a host-based access control list for deciding whether or not to accept connections from clients on a particular machine. This list initially consists of the host on which the server is running as well as any machines listed in the file */etc/X***n***.hosts*, where **n** is the display number of the server. Each line of the file should contain an Internet hostname (e.g. expo.lcs.mit.edu.) There should be no leading or trailing spaces on any lines. For example:

```
        joesworkstation
```

```
corporate.company.com
```

Users can add or remove hosts from this list and enable or disable access control using the *xhost* command from the same machine as the server. For example:

```
% xhost +janesworkstation
janesworkstation being added to access control list
% xhost +
all hosts being allowed (access control disabled)
% xhost -
all hosts being restricted (access control enabled)
% xhost
access control enabled (only the following hosts are allowed)
joesworkstation
janesworkstation
corporate.company.com
```

**SIGNALS**

The X server attaches special meaning to the following signals:

*SIGHUP*
This signal causes the server to close all existing connections, free all resources, and restore all defaults. It is sent by the display manager (*xdm* or *vuelogin*) whenever the main user's main application exits to force the server to clean up and prepare for the next user.

*SIGTERM*
This signal causes the server to exit cleanly.

*SIGUSR1*
This signal is used quite differently from either of the above. When the server starts, it checks to see if it has inherited SIGUSR1 as SIG_IGN instead of the usual SIG_DFL. In this case, the server sends a SIGUSR1 to its parent process after it has set up the various connection schemes. *Xdm* uses this feature to recognize when connecting to the server is possible.

**FONTS**

Fonts are usually stored as individual files in directories. The list of directories in which the server looks when trying to open a font is controlled by the *font path.* Although most sites will choose to have the server start up with the appropriate font path (using the *-fp* option mentioned above), it can be overridden using the *xset* program.

Font databases are created by running the *mkfontdir* or *stmkdirs* program in the directory containing the compiled versions of the fonts (*mkfontdir*) or font outlines (*stmkdirs*.) Whenever fonts are added to a directory, *mkfontdir* or *stmkdirs* should be rerun so that the server can find the new fonts. **If *mkfontdir* or *stmkdirs* is not run, the server will not be able to find any of the new fonts in the directory.**

In addition, the X server supports font servers. A font server is a networked program that supplies fonts to X servers and other capable programs. In order to communicate with a font server, the font servers address must be supplied as part of the X server's font path. A font server's address is specified as

*transport/hostname:port-number*

where *transport* is always *tcp*, *hostname* is the hostname of the machine being connected to (no hostname means a local connection) and *port-number* is the tcp address that the font server is listening at (typically 7000.)

**DIAGNOSTICS**

Too numerous to list them all. If run from *init(1M)*, errors are logged in the file */usr/adm/X*msgs*,

**FILES**

| | |
|---|---|
| /etc/inittab | Script for the init process |

| | |
|---|---|
| /etc/gettydefs | Speed and terminal settings used by getty |
| /etc/X*.hosts | Initial access control list |
| /usr/lib/X11/fonts | Top level font directory |
| /usr/lib/X11/rgb.txt | Color database |
| /usr/lib/X11/rgb.pag | Color database |
| /usr/lib/X11/rgb.dir | Color database |
| /usr/spool/sockets/X11/* | IPC mechanism socket |
| /usr/adm/X*msgs | Error log file |
| /usr/lib/X11/X*devices | Input devices used by the server. This file contains many example configurations. |
| /usr/lib/X11/X*screens | Screens used by the server. This file contains many example configurations. |
| /usr/lib/X11/X*pointerkeys | Keyboard pointer device file. This file contains many example configurations. |
| /usr/lib/X11/XHPkeymaps | Key device database used by the X server. |

A

**NOTES**

The option syntax is inconsistent with itself and *xset(1)*.

The acceleration option should take a numerator and a denominator like the protocol.

The color database is missing a large number of colors. However, there doesn't seem to be a better one available that can generate RGB values.

**COPYRIGHT**

**ORIGIN**

MIT Distribution

**SEE ALSO**

vuelogin(1X) bdftopcf(1) fs(1), getty(1M), gettydefs(4), gwindstop(1), hpterm(1), init(1M), inittab(4), mkfontdir(1), rgb(1), stmkdirs(1), x11start(1), xclock(1), xfd(1), xhost(1), xinit(1), xinitcolormap(1), xload(1), xmodmap(1), xrefresh(1), xseethru(1), xset(1), xsetroot(1), xterm(1), xwcreate(1), xwd(1), xwdestroy(1), xwininfo(1), xwud(1),

## NAME

xset - user preference utility for X

## SYNOPSIS

**xset** [-display *display*] [-b] [b on/off] [b [*volume* [*pitch* [*duration*]]] [[-]bc] [-c] [c on/off] [c [*volume*]] [[-+]fp[-+=] *path*[,*path*[,...]]] [fp default] [fp rehash] [[-]led [*integer*]] [led on/off] [m[ouse] [*accel_mult*[/*accel_div*] [*threshold*]]] [m[ouse] default] [p *pixel color*] [[-]r [*keycode*]] [r on/off] [s [*length* [*period*]]] [s blank/noblank] [s expose/noexpose] [s on/off] [s default] [q]

## DESCRIPTION

This program is used to set various user preference options of the display.

## OPTIONS

A

**-display** *display*
: This option specifies the server to use; see *X(1)*.

**b**
: The **b** option controls bell volume, pitch and duration. This option accepts up to three numerical parameters, a preceding dash(-), or a 'on/off' flag. If no parameters are given, or the 'on' flag is used, the system defaults will be used. If the dash or 'off' are given, the bell will be turned off. If only one numerical parameter is given, the bell volume will be set to that value, as a percentage of its maximum. Likewise, the second numerical parameter specifies the bell pitch, in hertz, and the third numerical parameter specifies the duration in milliseconds. Note that not all hardware can vary the bell characteristics. The X server will set the characteristics of the bell as closely as it can to the user's specifications.

**bc**
: The **bc** option controls *bug compatibility* mode in the server, if possible; a preceding dash(-) disables the mode, otherwise the mode is enabled. Various pre-R4 clients pass illegal values in some protocol requests, and pre-R4 servers did not correctly generate errors in these cases. Such clients, when run against an R4 server, will terminate abnormally or otherwise fail to operate correctly. Bug compatibility mode explicitly reintroduces certain bugs into the X server, so that many such clients can still be run. This mode should be used with care; new application development should be done with this mode disabled. The server must support the MIT-SUNDRY-NONSTANDARD protocol extension in order for this option to work.

**c**
: The **c** option controls key click. This option can take an optional value, a preceding dash(-), or an 'on/off' flag. If no parameter or the 'on' flag is given, the system defaults will be used. If the dash or 'off' flag is used, keyclick will be disabled. If a value from 0 to 100 is given, it is used to indicate volume, as a percentage of the maximum. The X server will set the volume to the nearest value that the hardware can support.

**fp=** *path,...*
: The **fp=** sets the font path to the entries given in the path argument. The entries are interpreted by the server, not by the client. Typically they are directory names or font server names, but the interpretation is server-dependent.

**fp default**
: The **default** argument causes the font path to be reset to the server's default.

**fp rehash**
: The **rehash** argument resets the font path to its current value, causing the server to reread the font databases in the current font path. This is generally only used when adding new fonts to a font directory (after running *mkfontdir* to recreate the font database).

**-fp** or **fp-**
: The **-fp** and **fp-** options remove elements from the current font path. They must be followed by a comma-separated list of entries.

**+fp** or **fp+**
: This **+fp** and **fp+** options prepend and append elements to the current font path, respectively. They must be followed by a comma-separated list of entries.

**led** The **led** option controls the keyboard LEDs. This controls the turning on or off of one or all of the LEDs. It accepts an optional integer, a preceding dash(-) or an 'on/off' flag. If no parameter or the 'on' flag is given, all LEDs are turned on. If a preceding dash or the flag 'off' is given, all LEDs are turned off. If a value between 1 and 32 is given, that LED will be turned on or off depending on the existence of a preceding dash. A common LED which can be controlled is the "Caps Lock" LED. "xset led 3" would turn led #3 on. "xset -led 3" would turn it off. The particular LED values may refer to different LEDs on different hardware.

**m** The **m** option controls the mouse parameters. The parameters for the mouse are 'acceleration' and 'threshold'. The acceleration can be specified as an integer, or as a simple fraction. The mouse, or whatever pointer the machine is connected to, will go 'acceleration' times as fast when it travels more than 'threshold' pixels in a short time. This way, the mouse can be used for precise alignment when it is moved slowly, yet it can be set to travel across the screen in a flick of the wrist when desired. One or both parameters for the **m** option can be omitted, but if only one is given, it will be interpreted as the acceleration. If no parameters or the flag 'default' is used, the system defaults will be set.

**p** The **p** option controls pixel color values. The parameters are the color map entry number in decimal, and a color specification. The root background colors may be changed on some servers by altering the entries for BlackPixel and WhitePixel. Although these are often 0 and 1, they need not be. Also, a server may choose to allocate those colors privately, in which case an error will be generated. The map entry must not be a read-only color, or an error will result.

**r** The **r** option controls the autorepeat for one or all keys. It accepts an optional keycode, a preceding dash(-) or an 'on/off' flag. If no parameter or the 'on' flag is given, autorepeat will be enabled for all keys. If a preceding dash or the flag 'off' is given, autorepeat will be disabled for all keys. If a keycode between 0 and 255 is given, autorepeat for the corresponding key will be turned on or off depending on the existence of a preceding dash.

s The s option lets you set the screen saver parameters. This option accepts up to two numerical parameters, a 'blank/noblank' flag, an 'expose/noexpose' flag, an 'on/off' flag, or the 'default' flag. If no parameters or the 'default' flag is used, the system will be set to its default screen saver characteristics. The 'on/off' flags simply turn the screen saver functions on or off. The 'blank' flag sets the preference to blank the video (if the hardware can do so) rather than display a background pattern, while 'noblank' sets the preference to display a pattern rather than blank the video. The 'expose' flag sets the preference to allow window exposures (the server can freely discard window contents), while 'noexpose' sets the preference to disable screen saver unless the server can regenerate the screens without causing exposure events. The length and period parameters for the screen saver function determines how long the server must be inactive for screen saving to activate, and the period to change the background pattern to avoid burn in. The arguments are specified in seconds. If only one numerical parameter is given, it will be used for the length.

**q** The **q** option gives you information on the current settings.

These settings will be reset to default values when you log out.

Note that not all X implementations are guaranteed to honor all of these options.

## SEE ALSO

X(1), Xserver(1), xmodmap(1), xrdb(1), xsetroot(1)

## COPYRIGHT

Copyright 1988, Massachusetts Institute of Technology.
See *X(1)* for a full statement of rights and permissions.

## AUTHOR

Bob Scheifler, MIT Laboratory for Computer Science
David Krikorian, MIT Project Athena (X11 version)

## NAME

xsetroot - root window parameter setting utility for X

## SYNOPSIS

**xsetroot** [-help] [-def] [-display *display*] [-cursor *cursorfile maskfile*] [-cursor_name *cursorname*] [-bitmap *filename*] [-mod *x y*] [-gray] [-grey] [-fg *color*] [-bg *color*] [-rv] [-solid *color*] [-name *string*]

## DESCRIPTION

The *setroot* program allows you to tailor the appearance of the background ("root") window on a workstation display running X. Normally, you experiment with *xsetroot* until you find a personalized look that you like, then put the *xsetroot* command that produces it into your X startup file. If no options are specified, or if *-def* is specified, the window is reset to its default state. The *-def* option can be specified along with other options and only the non-specified characteristics will be reset to the default state.

Only one of the background color/tiling changing options (-solid, -gray, -grey, -bitmap, and -mod) may be specified at a time.

A

## OPTIONS

The various options are as follows:

**-help** Print a usage message and exit.

**-def** Reset unspecified attributes to the default values. (Restores the background to the familiar gray mesh and the cursor to the hollow x shape.)

**-cursor** *cursorfile maskfile*
This lets you change the pointer cursor to whatever you want when the pointer cursor is outside of any window. Cursor and mask files are bitmaps (little pictures), and can be made with the *bitmap(1)* program. You probably want the mask file to be all black until you get used to the way masks work.

**-cursor_name** *cursorname*
This lets you change the pointer cursor to one of the standard cursors from the cursor font. Refer to appendix B of the X protocol for the names (except that the XC_ prefix is elided for this option).

**-bitmap** *filename*
Use the bitmap specified in the file to set the window pattern. You can make your own bitmap files (little pictures) using the *bitmap(1)* program. The entire background will be made up of repeated "tiles" of the bitmap.

**-mod** *x y*
This is used if you want a plaid-like grid pattern on your screen. x and y are integers ranging from 1 to 16. Try the different combinations. Zero and negative numbers are taken as 1.

**-gray** Make the entire background gray. (Easier on the eyes.)

**-grey** Make the entire background grey.

**-fg** *color*
Use "color" as the foreground color. Foreground and background colors are meaningful only in combination with -cursor, -bitmap, or -mod.

**-bg** *color*
Use "color" as the background color.

**-rv** This exchanges the foreground and background colors. Normally the foreground color is black and the background color is white.

**-solid** *color*
This sets the background of the root window to the specified color. This option is only useful on color servers.

**-name** *string*
Set the name of the root window to "string". There is no default value. Usually a name is

assigned to a window so that the window manager can use a text representation when the window is iconified. This option is unused since you can't iconify the background.

**-display** *display*
Specifies the server to connect to; see *X(1)*.

## SEE ALSO

X(1), xset(1), xrdb(1)

## COPYRIGHT

Copyright 1988, Massachusetts Institute of Technology.
See *X(1)* for a full statement of rights and permissions.

A

## AUTHOR

Mark Lillibridge, MIT Project Athena

A

**NAME**

xterm - terminal emulator for X

**SYNOPSIS**

**xterm** [-*toolkitoption* ...] [-option ...]

**DESCRIPTION**

The *xterm* program is a terminal emulator for the X Window System. It provides DEC VT102 and Tektronix 4014 compatible terminals for programs that can't use the window system directly. If the underlying operating system supports terminal resizing capabilities (for example, the SIGWINCH signal in systems derived from 4.3bsd), *xterm* will use the facilities to notify programs running in the window whenever it is resized.

The VT102 and Tektronix 4014 terminals each have their own window so that you can edit text in one and look at graphics in the other at the same time. To maintain the correct aspect ratio (height/width), Tektronix graphics will be restricted to the largest box with a 4014's aspect ratio that will fit in the window. This box is located in the upper left area of the window.

Although both windows may be displayed at the same time, one of them is considered the "active" window for receiving keyboard input and terminal output. This is the window that contains the text cursor. The active window can be chosen through escape sequences, the "VT Options" menu in the VT102 window, and the "Tek Options" menu in the 4014 window.

**EMULATIONS**

The VT102 emulation is fairly complete, but does not support the blinking character attribute nor the double-wide and double-size character sets. *Termcap*(5) entries that work with *xterm* include "xterm," "vt102," "vt100" and "ansi," and *xterm* automatically searches the termcap file in this order for these entries and then sets the "TERM" and the "TERMCAP" environment variables.

Many of the special *xterm* features (like logging) may be modified under program control through a set of escape sequences different from the standard VT102 escape sequences. (See the *Xterm Control Sequences* document.)

The Tektronix 4014 emulation is also fairly good. Four different font sizes and five different lines types are supported. The Tektronix text and graphics commands are recorded internally by *xterm* and may be written to a file by sending the COPY escape sequence (or through the **Tektronix** menu; see below). The name of the file will be "**COPY***yy-MM-dd.hh:mm:ss*", where *yy*, *MM*, *dd*, *hh*, *mm* and *ss* are the year, month, day, hour, minute and second when the COPY was performed (the file is created in the directory *xterm* is started in, or the home directory for a login *xterm*).

**OTHER FEATURES**

*Xterm* automatically highlights the text cursor when the pointer enters the window (selected) and unhighlights it when the pointer leaves the window (unselected). If the window is the focus window, then the text cursor is highlighted no matter where the pointer is.

In VT102 mode, there are escape sequences to activate and deactivate an alternate screen buffer, which is the same size as the display area of the window. When activated, the current screen is saved and replaced with the alternate screen. Saving of lines scrolled off the top of the window is disabled until the normal screen is restored. The *termcap*(5) entry for *xterm* allows the visual editor *vi*(1) to switch to the alternate screen for editing and to restore the screen on exit.

In either VT102 or Tektronix mode, there are escape sequences to change the name of the windows and to specify a new log file name. See *Xterm Control Sequences* for details. Enabling the escape sequence to change the log file name is a compile-time option; by default this escape sequence is ignored for security reasons.

**OPTIONS**

The *xterm* terminal emulator accepts all of the standard X Toolkit command line options as well as the following (if the option begins with a '+' instead of a '-', the option is restored to its default value):

**-help** This causes *xterm* to print out a verbose message describing its options.

**-132** Normally, the VT102 DECCOLM escape sequence that switches between 80 and 132 column mode is ignored. This option causes the DECCOLM escape sequence to be recognized, and the *xterm* window will resize appropriately.

**-ah** This option indicates that *xterm* should always highlight the text cursor. By default, *xterm* will display a hollow text cursor whenever the focus is lost or the pointer leaves the window.

**+ah** This option indicates that *xterm* should do text cursor highlighting based on focus.

**-b** *number*
This option specifies the size of the inner border (the distance between the outer edge of the characters and the window border) in pixels. The default is 2.

A

**-cb** Set the *vt100* resource **cutToBeginningOfLine** to FALSE.

**+cb** Set the *vt100* resource **cutToBeginningOfLine** to TRUE.

**-cc** *characterclassrange:value*[,...]
This sets classes indicated by the given ranges for using in selecting by words. See the section specifying character classes.

**-cn** This option indicates that newlines should not be cut in line-mode selections.

**+cn** This option indicates that newlines should be cut in line-mode selections.

**-cr** *color* This option specifies the color to use for text cursor. The default is to use the same foreground color that is used for text.

**-cu** This option indicates that *xterm* should work around a bug in the *curses*(3x) cursor motion package that causes the *more*(1) program to display lines that are exactly the width of the window and are followed by a line beginning with a tab to be displayed incorrectly (the leading tabs are not displayed).

**+cu** This option indicates that that *xterm* should not work around the *curses*(3x) bug mentioned above.

**-e** *program* [*arguments* ...]
This option specifies the program (and its command line arguments) to be run in the *xterm* window. It also sets the window title and icon name to be the basename of the program being executed if neither *-T* nor *-n* are given on the command line. **This must be the last option on the command line.**

**-fb** *font* This option specifies a font to be used when displaying bold text. This font must be the same height and width as the normal font. If only one of the normal or bold fonts is specified, it will be used as the normal font and the bold font will be produced by overstriking this font. The default is to do overstriking of the normal font.

**-im** Turn on the **useInsertMode** resource.

**+im** Turn off the **useInsertMode** resource.

**-j** This option indicates that *xterm* should do jump scrolling. Normally, text is scrolled one line at a time; this option allows *xterm* to move multiple lines at a time so that it doesn't fall as far behind. Its use is strongly recommended since it make *xterm* much faster when scanning through large amounts of text. The VT100 escape sequences for enabling and disabling smooth scroll as well as the "VT Options" menu can be used to turn this feature on or off.

**+j** This option indicates that *xterm* should not do jump scrolling.

**-l** This option indicates that *xterm* should send all terminal output to a log file as well as to the screen. This option can be enabled or disabled using the "VT Options" menu.

**+l** This option indicates that *xterm* should not do logging.

**-lf** *filename*
This option specifies the name of the file to which the output log described above is written. If *file* begins with a pipe symbol (|), the rest of the string is assumed to be a command to be used as the endpoint of a pipe. The ability to log to a pipe is a compile-time option which is disabled by default for security reasons. The default filename is "**XtermLog.***XXXXX*" (where *XXXXX* is the process id of *xterm*) and is created in the

directory from which *xterm* was started (or the user's home directory in the case of a login window).

**-ls** This option indicates that the shell that is started in the *xterm* window be a login shell (i.e. the first character of argv[0] will be a dash, indicating to the shell that it should read the user's .login or .profile).

**+ls** This option indicates that the shell that is started should not be a login shell (i.e. it will be a normal "subshell").

**-mb** This option indicates that *xterm* should ring a margin bell when the user types near the right end of a line. This option can be turned on and off from the "VT Options" menu.

**+mb** This option indicates that margin bell should not be rung.

**-mc milliseconds**
This option specifies the maximum time between multi-click selections.

**-ms** *color*
This option specifies the color to be used for the pointer cursor. The default is to use the foreground color.

**-nb** *number*
This option specifies the number of characters from the right end of a line at which the margin bell, if enabled, will ring. The default is 10.

**-rw** This option indicates that reverse-wraparound should be allowed. This allows the cursor to back up from the leftmost column of one line to the rightmost column of the previous line. This is very useful for editing long shell command lines and is encouraged. This option can be turned on and off from the "VT Options" menu.

**+rw** This option indicates that reverse-wraparound should not be allowed.

**-aw** This option indicates that auto-wraparound should be allowed. This allows the cursor to automatically wrap to the beginning of the next line when when it is at the rightmost position of a line and text is output.

**+aw** This option indicates that auto-wraparound should not be allowed.

**-s** This option indicates that *xterm* may scroll asynchronously, meaning that the screen does not have to be kept completely up to date while scrolling. This allows *xterm* to run faster when network latencies are very high and is typically useful when running across a very large internet or many gateways.

**+s** This option indicates that *xterm* should scroll synchronously.

**-sb** This option indicates that some number of lines that are scrolled off the top of the window should be saved and that a scrollbar should be displayed so that those lines can be viewed. This option may be turned on and off from the "VT Options" menu.

**+sb** This option indicates that a scrollbar should not be displayed.

**-sf** This option indicates that Sun Function Key escape codes should be generated for function keys.

**+sf** This option indicates that the standard escape codes should be generated for function keys.

**-si** This option indicates that output to a window should not automatically reposition the screen to the bottom of the scrolling region. This option can be turned on and off from the "VT Options" menu.

**+si** This option indicates that output to a window should cause it to scroll to the bottom.

**-sk** This option indicates that pressing a key while using the scrollbar to review previous lines of text should cause the window to be repositioned automatically in the normal position at the bottom of the scroll region.

**+sk** This option indicates that pressing a key while using the scrollbar should not cause the window to be repositioned.

**-sl** *number*
This option specifies the number of lines to save that have been scrolled off the top of the screen. The default is 64.

**-t** This option indicates that *xterm* should start in Tektronix mode, rather than in VT102 mode. Switching between the two windows is done using the "Options" menus.

**+t** This option indicates that *xterm* should start in VT102 mode.

**-tm** *string*
This option specifies a series of terminal setting keywords followed by the characters that should be bound to those functions, similar to the *stty* program. Allowable keywords include: intr, quit, erase, kill, eof, eol, swtch, start, stop, brk, susp, dsusp, rprnt, flush, weras, and lnext. Control characters may be specified as ^char (e.g. ^c or ^u) and ^? may be used to indicate delete.

A

**-tn** *name*
This option specifies the name of the terminal type to be set in the TERM environment variable. This terminal type must exist in the *termcap(5)* database and should have *li#* and *co#* entries.

**-ut** This option indicates that *xterm* shouldn't write a record into the the system log file */etc/utmp*.

**+ut** This option indicates that *xterm* should write a record into the system log file */etc/utmp*.

**-vb** This option indicates that a visual bell is preferred over an audible one. Instead of ringing the terminal bell whenever a Control-G is received, the window will be flashed.

**+vb** This option indicates that a visual bell should not be used.

**-wf** This option indicates that *xterm* should wait for the window to be mapped the first time before starting the subprocess so that the initial terminal size settings and environment variables are correct. It is the application's responsibility to catch subsequent terminal size changes.

**+wf** This option indicates that *xterm* show not wait before starting the subprocess.

**-C** This option indicates that this window should receive console output. This is not supported on all systems. To obtain console output, you must be the owner of the console device, and you must have read and write permission for it. If you are running X under *xdm* on the console screen you may need to have the session startup and reset programs explicitly change the ownership of the console device in order to get this option to work.

-*Sccn* This option specifies the last two letters of the name of a pseudoterminal to use in slave mode, plus the number of the inherited file descriptor. The option is parsed "%c%c%d". This allows *xterm* to be used as an input and output channel for an existing program and is sometimes used in specialized applications.

The following command line arguments are provided for compatibility with older versions. They may not be supported in the next release as the X Toolkit provides standard options that accomplish the same task.

*%geom* This option specifies the preferred size and position of the Tektronix window. It is shorthand for specifying the "**tekGeometry*" resource.

*#geom* This option specifies the preferred position of the icon window. It is shorthand for specifying the "**iconGeometry*" resource.

**-T** *string* This option specifies the title for *xterm*'s windows. It is equivalent to **-title**.

**-n** *string* This option specifies the icon name for *xterm*'s windows. It is shorthand for specifying the "**iconName*" resource. Note that this is not the same as the toolkit option **-name** (see below). The default icon name is the application name.

**-r** This option indicates that reverse video should be simulated by swapping the foreground and background colors. It is equivalent to **-rv**.

**-w** *number*
This option specifies the width in pixels of the border surrounding the window. It is equivalent to **-borderwidth** or **-bw**.

The following standard X Toolkit command line arguments are commonly used with *xterm*:

**-bg** *color*
This option specifies the color to use for the background of the window. The default is "white."

**-bd** *color*
This option specifies the color to use for the border of the window. The default is "black."

**-bw** *number*
This option specifies the width in pixels of the border surrounding the window.

**-fg** *color* This option specifies the color to use for displaying text. The default is "black."

**-fn** *font* This option specifies the font to be used for displaying normal text. The default is *fixed*.

**-name** *name*
This option specifies the application name under which resources are to be obtained, rather than the default executable file name. *Name* should not contain "." or "*" characters.

**-title** *string*
This option specifies the window title string, which may be displayed by window managers if the user so chooses. The default title is the command line specified after the **-e** option, if any, otherwise the application name.

**-rv** This option indicates that reverse video should be simulated by swapping the foreground and background colors.

**-geometry** *geometry*
This option specifies the preferred size and position of the VT102 window; see *X(1)*.

**-display** *display*
This option specifies the X server to contact; see *X(1)*.

**-xrm** *resourcestring*
This option specifies a resource string to be used. This is especially useful for setting resources that do not have separate command line options.

**-iconic** This option indicates that *xterm* should ask the window manager to start it as an icon rather than as the normal window.

## RESOURCES

The program understands all of the core X Toolkit resource names and classes as well as:

**iconGeometry** (class **IconGeometry**)
Specifies the preferred size and position of the application when iconified. It is not necessarily obeyed by all window managers.

**iconName** (class **IconName**)
Specifies the icon name. The default is the application name.

**termName** (class **TermName**)
Specifies the terminal type name to be set in the TERM environment variable.

**title** (class **Title**)
Specifies a string that may be used by the window manager when displaying this application.

**ttyModes** (class **TtyModes**)
Specifies a string containing terminal setting keywords and the characters to which they may be bound. Allowable keywords include: intr, quit, erase, kill, eof, eol, swtch, start, stop, brk, susp, dsusp, rprnt, flush, weras, and lnext. Control characters may be specified as ^char (e.g. ^c or ^u) and ^? may be used to indicate Delete. This is very useful for

overriding the default terminal settings without having to do an *stty* every time an *xterm* is started.

**useInsertMode** (class **UseInsertMode**)
Force use of insert mode by adding appropriate entries to the TERMCAP environment variable. This is useful if the system termcap is broken. The default is "false."

**utmpInhibit** (class **UtmpInhibit**)
Specifies whether or not *xterm* should try to record the user's terminal in */etc/utmp*.

A

**sunFunctionKeys** (class **SunFunctionKeys**)
Specifies whether or not Sun Function Key escape codes should be generated for function keys instead of standard escape sequences.

**waitForMap** (class **WaitForMap**)
Specifies whether or not *xterm* should wait for the initial window map before starting the subprocess. The default is "false."

The following resources are specified as part of the *vt100* widget (class *VT100*):

**allowSendEvents** (class **AllowSendEvents**)
Specifies whether or not synthetic key and button events (generated using the X protocol SendEvent request) should be interpreted or discarded. The default is "false" meaning they are discarded. Note that allowing such events creates a very large security hole.

**alwaysHighlight** (class **AlwaysHighlight**)
Specifies whether or not *xterm* should always display a highlighted text cursor. By default, a hollow text cursor is displayed whenever the pointer moves out of the window or the window loses the input focus.

**appcursorDefault** (class **AppcursorDefault**)
If "true," the cursor keys are initially in application mode. The default is "false."

**appkeypadDefault** (class **AppkeypadDefault**)
If "true," the keypad keys are initially in application mode. The default is "false."

**autoWrap** (class **AutoWrap**)
Specifies whether or not auto-wraparound should be enabled. The default is "true."

**bellSuppressTime** (class **BellSuppressTime**)
Number of milliseconds after a bell command is sent during which additional bells will be suppressed. Default is 200. If set non-zero, additional bells will also be suppressed until the server reports that processing of the first bell has been completed; this feature is most useful with the visible bell.

**boldFont** (class **BoldFont**)
Specifies the name of the bold font to use instead of overstriking.

**c132** (class **C132**)
Specifies whether or not the VT102 DECCOLM escape sequence should be honored. The default is "false."

**cutNewline** (class **CutNewline**)
If false, triple clicking to select a line does not include the Newline at the end of the line. If true, the Newline is selected. The default is "true."

**cutToBeginningOfLine** (class **CutToBeginningOfLine**)
If false, triple clicking to select a line selects only from the current word forward. If true, the entire line is selected. The default is "true."

**charClass** (class **CharClass**)
Specifies comma-separated lists of character class bindings of the form *[low-]high:value*. These are used in determining which sets of characters should be treated the same when doing cut and paste. See the section on specifying character classes.

**curses** (class **Curses**)
Specifies whether or not the last column bug in *curses*(3x) should be worked around. The default is "false."

**background** (class **Background**)
Specifies the color to use for the background of the window. The default is "white."

**foreground** (class **Foreground**)
Specifies the color to use for displaying text in the window. Setting the class name instead of the instance name is an easy way to have everything that would normally appear in the text color change color. The default is "black."

**cursorColor** (class **Foreground**)
Specifies the color to use for the text cursor. The default is "black."

**eightBitInput** (class **EightBitInput**)
If true, Meta characters input from the keyboard are presented as a single character with the eighth bit turned on. If false, Meta characters are converted into a two-character sequence with the character itself preceded by ESC. The default is "true."

**eightBitOutput** (class **EightBitOutput**)
Specifies whether or not eight-bit characters sent from the host should be accepted as is or stripped when printed. The default is "true."

**font** (class **Font**)
Specifies the name of the normal font. The default is "fixed."

**font1** (class **Font1**)
Specifies the name of the first alternative font.

**font2** (class **Font2**)
Specifies the name of the second alternative font.

**font3** (class **Font3**)
Specifies the name of the third alternative font.

**font4** (class **Font4**)
Specifies the name of the fourth alternative font.

**font5** (class **Font5**)
Specifies the name of the fifth alternative font.

**font6** (class **Font6**)
Specifies the name of the sixth alternative font.

**geometry** (class **Geometry**)
Specifies the preferred size and position of the VT102 window.

**internalBorder** (class **BorderWidth**)
Specifies the number of pixels between the characters and the window border. The default is 2.

**jumpScroll** (class **JumpScroll**)
Specifies whether or not jump scroll should be used. The default is "true."

**logFile** (class **Logfile**)
Specifies the name of the file to which a terminal session is logged. The default is "**XtermLog.***XXXXX*" (where *XXXXX* is the process id of *xterm*).

**logging** (class **Logging**)
Specifies whether or not a terminal session should be logged. The default is "false."

**logInhibit** (class **LogInhibit**)
Specifies whether or not terminal session logging should be inhibited. The default is "false."

**loginShell** (class **LoginShell**)
Specifies whether or not the shell to be run in the window should be started as a login shell. The default is "false."

**marginBell** (class **MarginBell**)
Specifies whether or not the bell should be run when the user types near the right margin. The default is "false."

**multiClickTime** (class **MultiClickTime**)
Specifies the maximum time in milliseconds between multi-click select events. The default is 250 milliseconds.

**multiScroll** (class **MultiScroll**)
Specifies whether or not scrolling should be done asynchronously. The default is "false."

**nMarginBell** (class **Column**)
Specifies the number of characters from the right margin at which the margin bell should be rung, when enabled.

A

**pointerColor** (class **Foreground**)
Specifies the foreground color of the pointer. The default is "XtDefaultForeground."

**pointerColorBackground** (class **Background**)
Specifies the background color of the pointer. The default is "XtDefaultBackground."

**pointerShape** (class **Cursor**)
Specifies the name of the shape of the pointer. The default is "xterm."

**resizeGravity** (class **ResizeGravity**)
Affects the behavior when the window is resized to be taller or shorter. **NorthWest** specifies that the top line of text on the screen stay fixed. If the window is made shorter, lines are dropped from the bottom; if the window is made taller, blank lines are added at the bottom. This is compatible with the behavior in R4. **SouthWest** (the default) specifies that the bottom line of text on the screen stay fixed. If the window is made taller, additional saved lines will be scrolled down onto the screen; if the window is made shorter, lines will be scrolled off the top of the screen, and the top saved lines will be dropped.

**reverseVideo** (class **ReverseVideo**)
Specifies whether or not reverse video should be simulated. The default is "false."

**reverseWrap** (class **ReverseWrap**)
Specifies whether or not reverse-wraparound should be enabled. The default is "false."

**saveLines** (class **SaveLines**)
Specifies the number of lines to save beyond the top of the screen when a scrollbar is turned on. The default is 64.

**scrollBar** (class **ScrollBar**)
Specifies whether or not the scrollbar should be displayed. The default is "false."

**scrollTtyOutput** (class **ScrollCond**)
Specifies whether or not output to the terminal should automatically cause the scrollbar to go to the bottom of the scrolling region. The default is "true."

**scrollKey** (class **ScrollCond**)
Specifies whether or not pressing a key should automatically cause the scrollbar to go to the bottom of the scrolling region. The default is "false."

**scrollLines** (class **ScrollLines**)
Specifies the number of lines that the *scroll-back* and *scroll-forw* actions should use as a default. The default value is 1.

**signalInhibit** (class **SignalInhibit**)
Specifies whether or not the entries in the "Main Options" menu for sending signals to *xterm* should be disallowed. The default is "false."

**tekGeometry** (class **Geometry**)
Specifies the preferred size and position of the Tektronix window.

**tekInhibit** (class **TekInhibit**)
Specifies whether or not Tektronix mode should be disallowed. The default is "false."

**tekSmall** (class **TekSmall**)
Specifies whether or not the Tektronix mode window should start in its smallest size if no explicit geometry is given. This is useful when running *xterm* on displays with small screens. The default is "false."

**tekStartup** (class **TekStartup**)
Specifies whether or not *xterm* should start up in Tektronix mode. The default is "false."

**titeInhibit** (class **TiteInhibit**)
Specifies whether or not *xterm* should remove remove *ti* and *te* termcap entries (used to switch between alternate screens on startup of many screen-oriented programs) from the TERMCAP string. If set, *xterm* also ignores the escape sequence to switch to the alternate screen.

**translations** (class **Translations**)
Specifies the key and button bindings for menus, selections, "programmed strings," etc. See **ACTIONS** below.

**visualBell** (class **VisualBell**)
Specifies whether or not a visible bell (i.e. flashing) should be used instead of an audible bell when Control-G is received. The default is "false."

The following resources are specified as part of the *tek4014* widget (class *Tek4014*):

**width** (class **Width**)
Specifies the width of the Tektronix window in pixels.

**height** (class **Height**)
Specifies the height of the Tektronix window in pixels.

**fontLarge** (class **Font**)
Specifies the large font to use in the Tektronix window.

**font2** (class **Font**)
Specifies font number 2 to use in the Tektronix window.

**font3** (class **Font**)
Specifies font number 3 to use in the Tektronix window.

**fontSmall** (class **Font**)
Specifies the small font to use in the Tektronix window.

**initialFont** (class **InitialFont**)
Specifies which of the four Tektronix fonts to use initially. Values are the same as for the *set-tek-text* action. The default is "large."

**ginTerminator** (class **GinTerminator**)
Specifies what character(s) should follow a GIN report or status report. The possibilities are "none," which sends no terminating characters, "CRonly," which sends CR, and "CR&EOT," which sends both CR and EOT. The default is "none."

The resources that may be specified for the various menus are described in the documentation for the Athena **SimpleMenu** widget. The name and classes of the entries in each of the menus are listed below.

The *mainMenu* has the following entries:

**securekbd** (class **SmeBSB**)
This entry invokes the **secure()** action.

**allowsends** (class **SmeBSB**)
This entry invokes the **allow-send-events(toggle)** action.

**logging** (class **SmeBSB**)
This entry invokes the **set-logging(toggle)** action.

**redraw** (class **SmeBSB**)
  This entry invokes the **redraw()** action.

**line1** (class **SmeLine**)
  This is a separator.

**suspend** (class **SmeBSB**)
  This entry invokes the **send-signal(tstp)** action on systems that support job control.

**continue** (class **SmeBSB**)
  This entry invokes the **send-signal(cont)** action on systems that support job control.

A

**interrupt** (class **SmeBSB**)
  This entry invokes the **send-signal(int)** action.

**hangup** (class **SmeBSB**)
  This entry invokes the **send-signal(hup)** action.

**terminate** (class **SmeBSB**)
  This entry invokes the **send-signal(term)** action.

**kill** (class **SmeBSB**)
  This entry invokes the **send-signal(kill)** action.

**line2** (class **SmeLine**)
  This is a separator.

**quit** (class **SmeBSB**)
  This entry invokes the **quit()** action.

The *vtMenu* has the following entries:

**scrollbar** (class **SmeBSB**)
  This entry invokes the **set-scrollbar(toggle)** action.

**jumpscroll** (class **SmeBSB**)
  This entry invokes the **set-jumpscroll(toggle)** action.

**reversevideo** (class **SmeBSB**)
  This entry invokes the **set-reverse-video(toggle)** action.

**autowrap** (class **SmeBSB**)
  This entry invokes the **set-autowrap(toggle)** action.

**reversewrap** (class **SmeBSB**)
  This entry invokes the **set-reversewrap(toggle)** action.

**autolinefeed** (class **SmeBSB**)
  This entry invokes the **set-autolinefeed(toggle)** action.

**appcursor** (class **SmeBSB**)
  This entry invokes the **set-appcursor(toggle)** action.

**appkeypad** (class **SmeBSB**)
  This entry invokes the **set-appkeypad(toggle)** action.

**scrollkey** (class **SmeBSB**)
  This entry invokes the **set-scroll-on-key(toggle)** action.

**scrollttyoutput** (class **SmeBSB**)
  This entry invokes the **set-scroll-on-tty-output(toggle)** action.

**allow132** (class **SmeBSB**)
  This entry invokes the **set-allow132(toggle)** action.

**cursesemul** (class **SmeBSB**)
  This entry invokes the **set-cursesemul(toggle)** action.

**visualbell** (class **SmeBSB**)
  This entry invokes the **set-visualbell(toggle)** action.

**marginbell** (class **SmeBSB**)
This entry invokes the **set-marginbell(toggle)** action.

**altscreen** (class **SmeBSB**)
This entry is currently disabled.

**line1** (class **SmeLine**)
This is a separator.

**softreset** (class **SmeBSB**)
This entry invokes the **soft-reset()** action.

**hardreset** (class **SmeBSB**)
This entry invokes the **hard-reset()** action.

**clearsavedlines** (class **SmeBSB**)"
This entry invokes the **clear-saved-lines()** action.

**line2** (class **SmeLine**)
This is a separator.

**tekshow** (class **SmeBSB**)
This entry invokes the **set-visibility(tek,toggle)** action.

**tekmode** (class **SmeBSB**)
This entry invokes the **set-terminal-type(tek)** action.

**vthide** (class **SmeBSB**)
This entry invokes the **set-visibility(vt,off)** action.

The *fontMenu* has the following entries:

**fontdefault** (class **SmeBSB**)
This entry invokes the **set-vt-font(d)** action.

**font1** (class **SmeBSB**)
This entry invokes the **set-vt-font(1)** action.

**font2** (class **SmeBSB**)
This entry invokes the **set-vt-font(2)** action.

**font3** (class **SmeBSB**)
This entry invokes the **set-vt-font(3)** action.

**font4** (class **SmeBSB**)
This entry invokes the **set-vt-font(4)** action.

**font5** (class **SmeBSB**)
This entry invokes the **set-vt-font(5)** action.

**font6** (class **SmeBSB**)
This entry invokes the **set-vt-font(6)** action.

**fontescape** (class **SmeBSB**)
This entry invokes the **set-vt-font(e)** action.

**fontsel** (class **SmeBSB**)
This entry invokes the **set-vt-font(s)** action.

The *tekMenu* has the following entries:

**tektextlarge** (class **SmeBSB**)
This entry invokes the **set-tek-text(l)** action.

**tektext2** (class **SmeBSB**)
This entry invokes the **set-tek-text(2)** action.

**tektext3** (class **SmeBSB**)
This entry invokes the **set-tek-text(3)** action.

**tektextsmall** (class **SmeBSB**)
    This entry invokes the **set-tek-text(s)** action.

**line1** (class **SmeLine**)
    This is a separator.

**tekpage** (class **SmeBSB**)
    This entry invokes the **tek-page()** action.

**tekreset** (class **SmeBSB**)
    This entry invokes the **tek-reset()** action.

A

**tekcopy** (class **SmeBSB**)
    This entry invokes the **tek-copy()** action.

**line2** (class **SmeLine**)
    This is a separator.

**vtshow** (class **SmeBSB**)
    This entry invokes the **set-visibility(vt,toggle)** action.

**vtmode** (class **SmeBSB**)
    This entry invokes the **set-terminal-type(vt)** action.

**tekhide** (class **SmeBSB**)
    This entry invokes the **set-visibility(tek,toggle)** action.

The following resources are useful when specified for the Athena Scrollbar widget:

**thickness** (class **Thickness**)
    Specifies the width in pixels of the scrollbar.

**background** (class **Background**)
    Specifies the color to use for the background of the scrollbar.

**foreground** (class **Foreground**)
    Specifies the color to use for the foreground of the scrollbar. The "thumb" of the scrollbar is a simple checkerboard pattern alternating pixels for foreground and background color.

## POINTER USAGE

Once the VT102 window is created, *xterm* allows you to select text and copy it within the same or other windows.

The selection functions are invoked when the pointer buttons are used with no modifiers, and when they are used with the "shift" key. The assignment of the functions described below to keys and buttons may be changed through the resource database; see **ACTIONS** below.

Pointer button one (usually left) is used to save text into the cut buffer. Move the cursor to beginning of the text, and then hold the button down while moving the cursor to the end of the region and releasing the button. The selected text is highlighted and is saved in the global cut buffer and made the PRIMARY selection when the button is released. Double-clicking selects by words. Triple-clicking selects by lines. Quadruple-clicking goes back to characters, etc. Multiple-click is determined by the time from button up to button down, so you can change the selection unit in the middle of a selection. If the key/button bindings specify that an X selection is to be made, *xterm* will leave the selected text highlighted for as long as it is the selection owner.

Pointer button two (usually middle) 'types' (pastes) the text from the PRIMARY selection, if any, otherwise from the cut buffer, inserting it as keyboard input.

Pointer button three (usually right) extends the current selection. (Without loss of generality, you can swap "right" and "left" everywhere in the rest of this paragraph.) If pressed while closer to the right edge of the selection than the left, it extends/contracts the right edge of the selection. If you contract the selection past the left edge of the selection, *xterm* assumes you really meant the left edge, restores the original selection, then extends/contracts the left edge of the selection. Extension starts in the selection unit mode that the last selection or extension was performed in; you can multiple-click to cycle through them.

By cutting and pasting pieces of text without trailing new lines, you can take text from several places in different windows and form a command to the shell, for example, or take output from a program and insert it into your favorite editor. Since the cut buffer is globally shared among different applications, you should regard it as a 'file' whose contents you know. The terminal emulator and other text programs should be treating it as if it were a text file, i.e., the text is delimited by new lines.

The scroll region displays the position and amount of text currently showing in the window (highlighted) relative to the amount of text actually saved. As more text is saved (up to the maximum), the size of the highlighted area decreases.

Clicking button one with the pointer in the scroll region moves the adjacent line to the top of the display window.

Clicking button three moves the top line of the display window down to the pointer position.

Clicking button two moves the display to a position in the saved text that corresponds to the pointer's position in the scrollbar.

Unlike the VT102 window, the Tektronix window dows not allow the copying of text. It does allow Tektronix GIN mode, and in this mode the cursor will change from an arrow to a cross. Pressing any key will send that key and the current coordinate of the cross cursor. Pressing button one, two, or three will return the letters 'l', 'm', and 'r', respectively. If the 'shift' key is pressed when a pointer button is pressed, the corresponding upper case letter is sent. To distinguish a pointer button from a key, the high bit of the character is set (but this is bit is normally stripped unless the terminal mode is RAW; see *tty*(4) for details).

## MENUS

*Xterm* has four menus, named *mainMenu*, *vtMenu*, *fontMenu*, and *tekMenu*. Each menu pops up under the correct combinations of key and button presses. Most menus are divided into two section, separated by a horizontal line. The top portion contains various modes that can be altered. A check mark appears next to a mode that is currently active. Selecting one of these modes toggles its state. The bottom portion of the menu are command entries; selecting one of these performs the indicated function.

The **xterm** menu pops up when the "control" key and pointer button one are pressed in a window. The *mainMenu* contains items that apply to both the VT102 and Tektronix windows. The **Secure Keyboard** mode is be used when typing in passwords or other sensitive data in an unsecure environment; see **SECURITY** below. Notable entries in the command section of the menu are the **Continue**, **Suspend**, **Interrupt**, **Hangup**, **Terminate** and **Kill** which sends the SIGCONT, SIGTSTP, SIGINT, SIGHUP, SIGTERM and SIGKILL signals, respectively, to the process group of the process running under *xterm* (usually the shell). The **Continue** function is especially useful if the user has accidentally typed CTRL-Z, suspending the process.

The *vtMenu* sets various modes in the VT102 emulation, and is popped up when the "control" key and pointer button two are pressed in the VT102 window. In the command section of this menu, the soft reset entry will reset scroll regions. This can be convenient when some program has left the scroll regions set incorrectly (often a problem when using VMS or TOPS-20). The full reset entry will clear the screen, reset tabs to every eight columns, and reset the terminal modes (such as wrap and smooth scroll) to their initial states just after *xterm* has finished processing the command line options.

The *fontMenu* sets the font used in the VT102 window. In addition to the default font and a number of alternatives that are set with resources, the menu offers the font last specified by the Set Font escape sequence (see the document *Xterm Control Sequences*) and the current selection as a font name (if the PRIMARY selection is owned).

The *tekMenu* sets various modes in the Tektronix emulation, and is popped up when the "control" key and pointer button two are pressed in the Tektronix window. The current font size is checked in the modes section of the menu. The **PAGE** entry in the command section clears the Tektronix window.

## SECURITY

X environments differ in their security consciousness. MIT servers, run under *xdm*, are capable of using a "magic cookie" authorization scheme that can provide a reasonable level of security for many people. If your server is only using a host-based mechanism to control access to the server (see *xhost(1)*), then if you enable access for a host and other users are also permitted to run clients on that same host, there is every possibility that someone can run an application that will use the basic services of the X protocol to snoop on your activities, potentially capturing a transcript of everything you type at the keyboard. This is of particular concern when you want to type in a password or other sensitive data. The best solution to this problem is to use a better authorization mechanism that host-based control, but a simple mechanism exists for protecting keyboard input in *xterm*.

A

The **xterm** menu (see MENUS above) contains a **Secure Keyboard** entry which, when enabled, ensures that all keyboard input is directed *only* to *xterm* (using the GrabKeyboard protocol request). When an application prompts you for a password (or other sensitive data), you can enable **Secure Keyboard** using the menu, type in the data, and then disable **Secure Keyboard** using the menu again. Only one X client at a time can secure the keyboard, so when you attempt to enable **Secure Keyboard** it may fail. In this case, the bell will sound. If the **Secure Keyboard** succeeds, the foreground and background colors will be exchanged (as if you selected the **Reverse Video** entry in the **Modes** menu); they will be exchanged again when you exit secure mode. If the colors do *not* switch, then you should be *very* suspicious that you are being spoofed. If the application you are running displays a prompt before asking for the password, it is safest to enter secure mode *before* the prompt gets displayed, and to make sure that the prompt gets displayed correctly (in the new colors), to minimize the probability of spoofing. You can also bring up the menu again and make sure that a check mark appears next to the entry.

**Secure Keyboard** mode will be disabled automatically if your xterm window becomes iconified (or otherwise unmapped), or if you start up a reparenting window manager (that places a title bar or other decoration around the window) while in **Secure Keyboard** mode. (This is a feature of the X protocol not easily overcome.) When this happens, the foreground and background colors will be switched back and the bell will sound in warning.

## CHARACTER CLASSES

Clicking the middle mouse button twice in rapid succession will cause all characters of the same class (e.g. letters, white space, punctuation) to be selected. Since different people have different preferences for what should be selected (for example, should filenames be selected as a whole or only the separate subnames), the default mapping can be overridden through the use of the *charClass* (class *CharClass*) resource.

This resource is simply a list of *range:value* pairs where the range is either a single number or *low-high* in the range of 0 to 127, corresponding to the ASCII code for the character or characters to be set. The *value* is arbitrary, although the default table uses the character number of the first character occurring in the set.

The default table is

```
static int charClass[128] = {
/* NUL  SOH  STX  ETX  EOT  ENQ  ACK  BEL */
    32,   1,   1,   1,   1,   1,   1,   1,

/*  BS   HT   NL   VT   NP   CR   SO   SI */
     1,  32,   1,   1,   1,   1,   1,   1,

/* DLE  DC1  DC2  DC3  DC4  NAK  SYN  ETB */
     1,   1,   1,   1,   1,   1,   1,   1,

/* CAN   EM  SUB  ESC   FS   GS   RS   US */
     1,   1,   1,   1,   1,   1,   1,   1,

/*  SP    !    "    #    $    %    &    ' */
    32,  33,  34,  35,  36,  37,  38,  39,
/*   (    )    *    +    ,    -    .    / */
    40,  41,  42,  43,  44,  45,  46,  47,
/*   0    1    2    3    4    5    6    7 */
    48,  48,  48,  48,  48,  48,  48,  48,
/*   8    9    :    ;    <    =    >    ? */
    48,  48,  58,  59,  60,  61,  62,  63,
/*   @    A    B    C    D    E    F    G */
    64,  48,  48,  48,  48,  48,  48,  48,
/*   H    I    J    K    L    M    N    O */
    48,  48,  48,  48,  48,  48,  48,  48,
/*   P    Q    R    S    T    U    V    W */
    48,  48,  48,  48,  48,  48,  48,  48,
/*   X    Y    Z    [    \    ]    ^    _ */
    48,  48,  48,  91,  92,  93,  94,  48,
/*   `    a    b    c    d    e    f    g */
    96,  48,  48,  48,  48,  48,  48,  48,
/*   h    i    j    k    l    m    n    o */
    48,  48,  48,  48,  48,  48,  48,  48,
/*   p    q    r    s    t    u    v    w */
    48,  48,  48,  48,  48,  48,  48,  48,
/*   x    y    z    {    |    }    ~  DEL */
    48,  48,  48, 123, 124, 125, 126,   1};
```

For example, the string "33:48,37:48,45-47:48,64:48" indicates that the exclamation mark, percent sign, dash, period, slash, and ampersand characters should be treated the same way as characters and numbers. This is very useful for cutting and pasting electronic mailing addresses and filenames.

## ACTIONS

It is possible to rebind keys (or sequences of keys) to arbitrary strings for input, by changing the translations for the vt100 or tek4014 widgets. Changing the translations for events other than key and button events is not expected, and will cause unpredictable behavior. The following actions are provided for using within the *vt100* or *tek4014* **translations** resources:

**bell([*percent*])**
: This action rings the keyboard bell at the specified percentage above or below the base volume.

**ignore()** This action ignores the event but checks for special pointer position escape sequences.

**insert()** This action inserts the character or string associated with the key that was pressed.

**insert-seven-bit()**
: This action is a synonym for **insert()**

**insert-eight-bit()**
: This action inserts an eight-bit (Meta) version of the character or string associated with the key that was pressed. The exact action depends on the value of the **eightBitInput** resource.

**insert-selection**(*sourcename* [, ...])
: This action inserts the string found in the selection or cutbuffer indicated by *sourcename*. Sources are checked in the order given (case is significant) until one is found. Commonly-used selections include: *PRIMARY*, *SECONDARY*, and *CLIPBOARD*. Cut buffers are typically named *CUT_BUFFER0* through *CUT_BUFFER7*.

**keymap**(*name*)
: This action dynamically defines a new translation table whose resource name is *name* with the suffix *Keymap* (case is significant). The name *None* restores the original translation table.

A

**popup-menu**(*menuname*)
: This action displays the specified popup menu. Valid names (case is significant) include: *mainMenu*, *vtMenu*, *fontMenu*, and *tekMenu*.

**secure**() This action toggles the *Secure Keyboard* mode described in the section named **SECURITY**, and is invoked from the **securekbd** entry in *mainMenu*.

**select-start**()
: This action begins text selection at the current pointer location. See the section on **POINTER USAGE** for information on making selections.

**select-extend**()
: This action tracks the pointer and extends the selection. It should only be bound to Motion events.

**select-end**(*destname* [, ...])
: This action puts the currently selected text into all of the selections or cutbuffers specified by *destname*.

**select-cursor-start**()
: This action is similar to **select-start** except that it begins the selection at the current text cursor position.

**select-cursor-end**(*destname* [, ...])
: This action is similar to **select-end** except that it should be used with **select-cursor-start**.

**set-vt-font**(*d/1/2/3/4/5/6/e/s* [,*normalfont* [, *boldfont*]])
: This action sets the font or fonts currently being used in the VT102 window. The first argument is a single character that specifies the font to be used: *d* or *D* indicate the default font (the font initially used when *xterm* was started), *1* through *6* indicate the fonts specified by the *font1* through *font6* resources, *e* or *E* indicate the normal and bold fonts that have been set through escape codes (or specified as the second and third action arguments, respectively), and *s* or *S* indicate the font selection (as made by programs such as *xfontsel(1)*) indicated by the second action argument.

**start-extend**()
: This action is similar to **select-start** except that the selection is extended to the current pointer location.

**start-cursor-extend**()
: This action is similar to **select-extend** except that the selection is extended to the current text cursor position.

**string**(*string*)
: This action inserts the specified text string as if it had been typed. Quotation is necessary if the string contains whitespace or non-alphanumeric characters. If the string argument begins with the characters "0x", it is interpreted as a hex character constant.

**scroll-back**(*count* [,*units*])
: This action scrolls the text window backward so that text that had previously scrolled off the top of the screen is now visible. The *count* argument indicates the number of *units* (which may be *page*, *halfpage*, *pixel*, or *line*) by which to scroll.

**scroll-forw**(*count* [,*units*])
: This action scrolls is similar to **scroll-back** except that it scrolls the other direction.

**allow-send-events**(*on/off/toggle*)
: This action set or toggles the **allowSendEvents** resource and is also invoked by the **allowsends** entry in *mainMenu.*

**set-logging**(*on/off/toggle*)
: This action toggles the **logging** resource and is also invoked by the **logging** entry in *mainMenu.*

**redraw()**
: This action redraws the window and is also invoked by the *redraw* entry in *mainMenu.*

**send-signal**(*signame*)
: This action sends the signal named by *signame* to the *xterm* subprocess (the shell or program specified with the *-e* command line option) and is also invoked by the **suspend**, **continue**, **interrupt**, **hangup**, **terminate**, and *kill* entries in *mainMenu.* Allowable signal names are (case is not significant): *tstp* (if supported by the operating system), *suspend* (same as *tstp*), *cont* (if supported by the operating system), *int*, *hup*, *term*, *quit*, *alrm*, *alarm* (same as *alrm*) and *kill*.

**quit()**
: This action sends a SIGHUP to the subprogram and exits. It is also invoked by the **quit** entry in *mainMenu.*

**set-scrollbar**(*on/off/toggle*)
: This action toggles the **scrollbar** resource and is also invoked by the **scrollbar** entry in *vtMenu.*

**set-jumpscroll**(*on/off/toggle*)
: This action toggles the **jumpscroll** resource and is also invoked by the **jumpscroll** entry in *vtMenu.*

**set-reverse-video**(*on/off/toggle*)
: This action toggles the *reverseVideo* resource and is also invoked by the **reversevideo** entry in *vtMenu.*

**set-autowrap**(*on/off/toggle*)
: This action toggles automatic wrapping of long lines and is also invoked by the **autowrap** entry in *vtMenu.*

**set-reversewrap**(*on/off/toggle*)
: This action toggles the **reverseWrap** resource and is also invoked by the **reversewrap** entry in *vtMenu.*

**set-autolinefeed**(*on/off/toggle*)
: This action toggles automatic insertion of linefeeds and is also invoked by the **autolinefeed** entry in *vtMenu.*

**set-appcursor**(*on/off/toggle*)
: This action toggles the handling Application Cursor Key mode and is also invoked by the Bappcursor entry in *vtMenu.*

**set-appkeypad**(*on/off/toggle*)
: This action toggles the handling of Application Keypad mode and is also invoked by the **appkeypad** entry in *vtMenu.*

**set-scroll-on-key**(*on/off/toggle*)
: This action toggles the **scrollKey** resource and is also invoked from the **scrollkey** entry in *vtMenu.*

**set-scroll-on-tty-output**(*on/off/toggle*)
: This action toggles the **scrollTtyOutput** resource and is also invoked from the **scrolltyoutput** entry in *vtMenu.*

**set-allow132**(*on/off/toggle*)
: This action toggles the **c132** resource and is also invoked from the **allow132** entry in *vtMenu.*

**set-cursesemul**(*on/off/toggle*)
This action toggles the **curses** resource and is also invoked from the **cursesemul** entry in *vtMenu*.

**set-visual-bell**(*on/off/toggle*)
This action toggles the **visualBell** resource and is also invoked by the **visualbell** entry in *vtMenu*.

**set-marginbell**(*on/off/toggle*)
This action toggles the **marginBell** resource and is also invoked from the **marginbell** entry in *vtMenu*.

A

**set-altscreen**(*on/off/toggle*)
This action toggles between the alternate and current screens.

**soft-reset**()
This action resets the scrolling region and is also invoked from the **softreset** entry in *vtMenu*.

**hard-reset**()
This action resets the scrolling region, tabs, window size, and cursor keys and clears the screen. It is also invoked from the **hardreset** entry in *vtMenu*.

**clear-saved-lines**()
This action does **hard-reset**() (see above) and also clears the history of lines saved off the top of the screen. It is also invoked from the **clearsavedlines** entry in *vtMenu*.

**set-terminal-type**(*type*)
This action directs output to either the *vt* or *tek* windows, according to the *type* string. It is also invoked by the **tekmode** entry in *vtMenu* and the **vtmode** entry in *tekMenu*.

**set-visibility**(*vt/tek,on/off/toggle*)
This action controls whether or not the *vt* or *tek* windows are visible. It is also invoked from the **tekshow** and **vthide** entries in *vtMenu* and the **vtshow** and **tekhide** entries in *tek-Menu*.

**set-tek-text**(*large/2/3/small*)
This action sets font used in the Tektronix window to the value of the resources **tek-textlarge**, **tektext2**, **tektext3**, and **tektextsmall** according to the argument. It is also by the entries of the same names as the resources in *tekMenu*.

**tek-page**()
This action clears the Tektronix window and is also invoked by the **tekpage** entry in *tek-Menu*.

**tek-reset**()
This action resets the Tektronix window and is also invoked by the *tekreset* entry in *tek-Menu*.

**tek-copy**()
This action copies the escape codes used to generate the current window contents to a file in the current directory beginning with the name COPY. It is also invoked from the *tekcopy* entry in *tekMenu*.

**visual-bell**()
This action flashes the window quickly.

The Tektronix window also has the following action:

**gin-press**(*l/L/m/M/r/R*)
This action sends the indicated graphics input code.

The default bindings in the VT102 window are:

```
         Shift <KeyPress> Prior:          scroll-back(1,halfpage) \n\
          Shift <KeyPress> Next:          scroll-forw(1,halfpage) \n\
        Shift <KeyPress> Select:          select-cursor-start() \
                                        select-cursor-end(PRIMARY, CUT_BUFFER0) \n\
       Shift <KeyPress> Insert:         insert-selection(PRIMARY, CUT_BUFFER0) \n\
            ~Meta<KeyPress>:            insert-seven-bit() \n\
             Meta<KeyPress>:            insert-eight-bit() \n\
           !Ctrl <Btn1Down>:            popup-menu(mainMenu) \n\
      !Lock Ctrl <Btn1Down>:            popup-menu(mainMenu) \n\
           ~Meta <Btn1Down>:            select-start() \n\
         ~Meta <Btn1Motion>:            select-extend() \n\
           !Ctrl <Btn2Down>:            popup-menu(vtMenu) \n\
      !Lock Ctrl <Btn2Down>:            popup-menu(vtMenu) \n\
     ~Ctrl ~Meta <Btn2Down>:            ignore() \n\
       ~Ctrl ~Meta <Btn2Up>:            insert-selection(PRIMARY, CUT_BUFFER0) \n\
           !Ctrl <Btn3Down>:            popup-menu(fontMenu) \n\
      !Lock Ctrl <Btn3Down>:            popup-menu(fontMenu) \n\
     ~Ctrl ~Meta <Btn3Down>:            start-extend() \n\
         ~Meta <Btn3Motion>:            select-extend() \n\
                    <BtnUp>:            select-end(PRIMARY, CUT_BUFFER0) \n\
                  <BtnDown>:            bell(0)
```

A

The default bindings in the Tektronix window are:

```
            ~Meta<KeyPress>:            insert-seven-bit() \n\
             Meta<KeyPress>:            insert-eight-bit() \n\
           !Ctrl <Btn1Down>:            popup-menu(mainMenu) \n\
      !Lock Ctrl <Btn1Down>:            popup-menu(mainMenu) \n\
           !Ctrl <Btn2Down>:            popup-menu(tekMenu) \n\
      !Lock Ctrl <Btn2Down>:            popup-menu(tekMenu) \n\
      Shift ~Meta<Btn1Down>:            gin-press(L) \n\
            ~Meta<Btn1Down>:            gin-press(l) \n\
      Shift ~Meta<Btn2Down>:            gin-press(M) \n\
            ~Meta<Btn2Down>:            gin-press(m) \n\
      Shift ~Meta<Btn3Down>:            gin-press(R) \n\
            ~Meta<Btn3Down>:            gin-press(r)
```

Below is a sample how of the **keymap()** action is used to add special keys for entering commonly-typed works:

```
*VT100.Translations: #override <Key>F13: keymap(dbx)
*VT100.dbxKeymap.translations: \
          <Key>F14:      keymap(None) \n\
          <Key>F17:      string("next") string(0x0d) \n\
          <Key>F18:      string("step") string(0x0d) \n\
          <Key>F19:      string("continue") string(0x0d) \n\
          <Key>F20:      string("print ") insert-selection(PRIMARY, CUT_BUFFER0)
```

## ENVIRONMENT

*Xterm* sets the environment variables "TERM" and "TERMCAP" properly for the size window you have created. It also uses and sets the environment variable "DISPLAY" to specify which bit map display terminal to use. The environment variable "WINDOWID" is set to the X window id number of the *xterm* window.

## SEE ALSO

resize(1), X(1), pty(4), tty(4)
*Xterm Control Sequences* (in the *xterm* source directory)

## BUGS

Large pastes do not work on some systems. This is not a bug in *xterm*; it is a bug in the pseudo terminal driver of those systems. *xterm* feeds large pastes to the pty only as fast as the pty will accept data, but some pty drivers do not return enough information to know if the write has succeeded.

Many of the options are not resettable after *xterm* starts.

The Tek widget does not support key/button re-binding.

Only fixed-width, character-cell fonts are supported.

A

This program still needs to be rewritten. It should be split into very modular sections, with the various emulators being completely separate widgets that don't know about each other. Ideally, you'd like to be able to pick and choose emulator widgets and stick them into a single control widget.

There needs to be a dialog box to allow entry of log file name and the COPY file name.

**COPYRIGHT**

**AUTHORS**

Far too many people, including:

Loretta Guarino Reid (DEC-UEG-WSL), Joel McCormack (DEC-UEG-WSL), Terry Weissman (DEC-UEG-WSL), Edward Moy (Berkeley), Ralph R. Swick (MIT-Athena), Mark Vandevoorde (MIT-Athena), Bob McNamara (DEC-MAD), Jim Gettys (MIT-Athena), Bob Scheifler (MIT X Consortium), Doug Mink (SAO), Steve Pitschke (Stellar), Ron Newman (MIT-Athena), Jim Fulton (MIT X Consortium), Dave Serisky (HP), Jonathan Kamens (MIT-Athena)

## NAME

xwcreate - create a new X window

## SYNOPSIS

**xwcreate** [options] name

## DESCRIPTION

This command creates a new X window and assigns it the name *name.*

This program will also create a device file of the same name as the window in the indicated directory. After the window has been created, the device file may be used to specify the window that an application should use when utilizing a graphics library (e.g., Starbase or HP-GKS).

A window created by *xwcreate* can be destroyed by *xwdestroy(1).*

## OPTIONS

**-display** *display*
Specifies the server to connect to; See *X(1)* for details. A limitation in xwcreate requires that the display name must be no more than 115 characters long.

**-parent** *parent*
Name of the window which is to be the parent of *name.* If named, the parent window must have been created by a previous invocation of *xwcreate* and must not have been destroyed by *xwdestroy(1);* otherwise an error message will be generated. If parent window is not named, the RootWindow of the display and screen will be used as the parent. If specified, parent window's name must be no more than 12 characters long.

**-geometry** *geometry*
This option specifies the preferred size and position of the window; See *X(1)* for details.

**-r**
Requests the X server to create backing store for the window. By default, windows are not created with backing store.

**-bg** *color*
This option specifies the background color. By default, background color of the window will be black.

**-bw** *pixels*
This option specifies the width in pixels of the window border. By default, border of the window will be 3 pixels wide.

**-bd** *color*
This option specifies the border color. By default, the window border will be white.

**-depth** *depth*
This option specifies the visual depth of the window. By default, the window will have the same depth as its parent. If the specified depth is not supported by the display, an error will be generated and the window will not be created.

**-visual** *visualclass*
This option specifies the visual class of the window when multiple visual classes are supported by the display at the specified depth. *visualclass* can be "PseudoColor", "StaticColor", "GrayScale", "StaticGray", "DirectColor" or "TrueColor" (case is not significant). For most displays, which support only one visual class at each depth, this option need not be specified.

**-overlay**
This option specifies that an overlay plane visual should be used. This option is only in effect for servers that are in the combined mode with the root window property, SERVER_OVERLAY_VISUALS set. For such a server, specifying this option will cause only overlay visuals to be considered; when this option is not specified, image plane based visuals will be favored.

**-wmdir** *directory*
is the name of the directory where the device file is to be created. See DEPENDENCIES, below, for details.

**-title** *name*
: is the name to be used to reference the window. The *name* must be no more than 12 characters long.

## X DEFAULTS

*xwcreate* uses the Xlib routine *XGetDefault(3X)* to read its Xdefaults, so its resource names are all capitalized.

**Background**
: Specifies the window's background color.

A

**BorderColor**
: Specifies the border color. This option is useful only on color displays.

**BorderWidth**
: Specifies the border width.

**Depth** Specifies the visual depth of the created window.

**VisualClass**
: Specifies the visual class of the created window.

**Retained**
: If 'on', requests the X server to create backing store for the window.

**Wmdir** Specifies the default directory where the device file will be created. See *-wmdir* above for details.

**Geometry**
: Specifies the default positioning and/or sizing for the created window. See *X(1)* for details.

## EXAMPLES

xwcreate FullView
: Create a window named "FullView". Since no other argument is provided, the default geometry, border color, etc. of FullView will be taken from the RootWindow of the window's display and screen.

xwcreate HalfView -display remote_host:1.2 -parent FullView
-geometry 400x200+5+10 -r -bw 10
: Create a window named "HalfView" on the display "remote_host:1.2". HalfView will be a child of the window "FullView". The upper left hand corner of HalfView will be located at coordinate 5,10 of FullView and will be 400 pixels wide and 200 pixels high. The border of HalfView will be 10 pixels wide and the border colors will be the same as FullView.

## DEPENDENCIES

HP-UX Systems

Windows are actually created, maintained, and destroyed by the *gwind* daemon; *xwcreate* does its job by requesting window creation from the daemon. If *gwind* is not running, *xwcreate* will start it.

The device file created by *xwcreate/gwind* is a *pty*, through which graphics applications (such as the Starbase library) communicate with *gwind*.

The location of the device file is determined by the use of the *-wmdir* command line option and the WMDIR environment variable.

If the *-wmdir* option is not used, then the directory name will be computed as follows: first, the environment of the process will be searched for the variable $WMDIR. If the variable $WMDIR is defined in the environment, then it will be used as the desired directory. If the the variable $WMDIR is not defined in the environment, then the device file will be created in the /dev/screen directory.

If the *-wmdir* option is used in the command line, the directory name will be obtained as follows: If the directory argument implies an absolute pathname, then it will be taken to be the desired directory. Otherwise, the directory name will be taken to be relative to the value of the environment variable $WMDIR. If $WMDIR is not defined in the environment, the directory name will be taken to be relative to the /dev/screen directory. Note: if $WMDIR is defined in the environment, it must represent an absolute pathname. If *-wmdir* is defined in the command line, then the implied directory must have already been created. Otherwise, an error ("Invalid directory") will be generated.

If *-wmdir* is used or *WMDIR* environment variable is set to other than the default, the directory used must exist on the same physical device as "/dev".

If XKillClient is used (used by some window managers) on one of the windows created by *xwcreate*, all windows created by *xwcreate* (started with the same "-display" argument) are also be destroyed.

**ENVIRONMENT**

DISPLAY - the default host and display number.
WMDIR - the window manager directory.
/dev/screen - the default window manager directory on HP-UX systems.

**DIAGNOSTICS**

If the window is created successfully, *xwcreate* will remain silent. Otherwise *xwcreate* prints one or more error messages to standard output. For example:

```
No such display.
Named window exists.
Named parent window does not exist.
Couldn't communicate with gwind.
```

**NOTES**

The WM_CLASS of an xwcreate'ed window is **Xwcreate.**

**ORIGIN**

HP

**SEE ALSO**

X(1), XOpenDisplay(3x), xwdestroy(1).

## NAME

xwd - dump an image of an X window

## SYNOPSIS

**xwd** [-debug] [-help] [-nobdrs] [-out *file*] [-xy] [-frame] [-add *value*] [-root | -id *id* | -name *name* ] [-icmap] [-screen] [-display *display*]

## DESCRIPTION

*Xwd* is an X Window System window dumping utility. *Xwd* allows X users to store window images in a specially formatted dump file. This file can then be read by various other X utilities for redisplay, printing, editing, formatting, archiving, image processing, etc. The target window is selected by clicking the pointer in the desired window. The keyboard bell is rung once at the beginning of the dump and twice when the dump is completed.

A

## OPTIONS

**-display** *display*
This argument allows you to specify the server to connect to; see *X(1)*.

**-help** Print out the 'Usage:' command syntax summary.

**-nobdrs** This argument specifies that the window dump should not include the pixels that compose the X window border. This is useful in situations where you may wish to include the window contents in a document as an illustration.

**-out** *file* This argument allows the user to explicitly specify the output file on the command line. The default is to output to standard out.

**-xy** This option applies to color displays only. It selects 'XY' format dumping instead of the default 'Z' format.

**-add** *value*
This option specifies an signed value to be added to every pixel.

**-frame** This option indicates that the window manager frame should be included when manually selecting a window.

**-root** This option indicates that the root window should be selected for the window dump, without requiring the user to select a window with the pointer.

**-id** *id* This option indicates that the window with the specified resource id should be selected for the window dump, without requiring the user to select a window with the pointer.

**-name** *name*
This option indicates that the window with the specified WM_NAME property should be selected for the window dump, without requiring the user to select a window with the pointer.

**-icmap** Normally the colormap of the chosen window is used to obtain RGB values. This option forces the first installed colormap of the screen to be used instead.

**-screen** This option indicates that the GetImage request used to obtain the image should be done on the root window, rather than directly on the specified window. In this way, you can obtain pieces of other windows that overlap the specified window, and more importantly, you can capture menus or other popups that are independent windows but appear over the specified window.

## ENVIRONMENT

**DISPLAY**
To get default host and display number.

## FILES

**XWDFile.h**
X Window Dump File format definition file.

## SEE ALSO

xwud(1), xpr(1), X(1)

## COPYRIGHT

Copyright 1988, Massachusetts Institute of Technology.

See *X(1)* for a full statement of rights and permissions.

**AUTHORS**

Tony Della Fera, Digital Equipment Corp., MIT Project Athena
William F. Wyatt, Smithsonian Astrophysical Observatory

**NAME**

xwd2sb - translate xwd bitmap to Starbase bitmap format

**SYNOPSIS**

**xwd2sb**

**DESCRIPTION**

This command translates a bitmap file created by the *xwd(1)* X window dump utility program into a Starbase bitmap file as described in *bitmapfile(4)*. Translation is done from standard input to standard output.

A

Bitmaps created by *xwd* in the *XYPixmap* format are translated into *plane-major full-depth* Starbase bitmaps. *ZPixmap* format bitmaps are translated into *pixel-major* Starbase bitmaps.

*Xwd* format bitmaps with visual class *TrueColor* or *DirectColor* are translated into Starbase bitmaps with the colormap mode *CMAP_FULL.* Other visual classes result in Starbase bitmaps with the *CMAP_NORMAL* colormap mode.

Window borders stored by *xwd* are stripped from the image during translation.

**OPTIONS**

none

**EXAMPLES**

```
xwd | xwd2sb | pcltrans | lp -oraw
```

Invokes *xwd* to dump the contents of a window in *ZPixmap* format, *xwd2sb* translates the window image into Starbase format, *pcltrans* prepares the image for printing, and *lp* spools the image for the printer.

```
xwd -xy | xwd2sb > sbimage
```

Invokes *xwd* to dump the contents of a window in *XYPixmap* format and *xwd2sb* to translate the image into Starbase plane-major full-depth format. The Starbase bitmap image is placed in the *sbimage* file. (Note that *pcltrans* is unable to process plane-major full-depth images.)

```
xwd2sb <xwdfile >sbfile
```

Translates the image in *xwdfile* to Starbase format and places the result in *sbfile.*

**RESTRICTIONS**

XWD bitmaps must be 1-8, 12, or 24 planes deep. Bitmaps of depth 1-8 may have a visual class of GrayScale, StaticGray, PseudoColor, or StaticColor. Bitmaps of depths 12 or 24 must be of the DirectColor or TrueColor visual class.

A 12 plane bitmap must have four bits each for red, green, and blue. A 24 plane bitmap must have eight bits each for red, green, and blue.

**ORIGIN**

Hewlett-Packard GTD

**SEE ALSO**

xwd(1), pcltrans(1), bitmapfile(4).

*Starbase Graphics Techniques, HP-UX Concepts and Tutorials,* chapters on "Color" and "Storing and Printing Images".

## NAME

xwdestroy - destroy one or more existing windows

## SYNOPSIS

**xwdestroy [-wmdir** *directory***]** *window1 window2* **...**

## DESCRIPTION

If a window named in the list was created using *xwcreate*(1), then it is destroyed, along with its children. Also the device files associated with these windows are removed.

**-wmdir** *directory*
: is the name of the directory where the device file for the window was created. See the *xwcreate*(1) man page for a description of how this option and the WMDIR environment variable are used to locate the device file.

A

## ENVIRONMENT

WMDIR - the window manager directory.
/dev/screen - the default window manager directory.

## DIAGNOSTICS

If the windows were destroyed successfully, the program remains silent. If one or more of the windows could not be destroyed because of some error, appropriate message will be printed on standard output. For example:

```
Invalid directory
Named window does not exist.
```

## ORIGIN

HP

## SEE ALSO

XOpenDisplay(3), xwcreate(1).

## NAME

xwininfo - window information utility for X

## SYNOPSIS

**xwininfo** [-help] [-id *id*] [-root] [-name *name*] [-int] [-children] [-tree] [-stats] [-bits] [-events] [-size] [-wm] [-shape] [-frame] [-all] [-english] [-metric] [-display *display*]

## DESCRIPTION

NOTE: This client is not supported by HP in HP-UX Release 9.0. It is included in the release, but it is now installed in the directory /usr/contrib/bin/X11. Your $PATH environment variable should contain this directory path in order to run this client. See the release notes For more information.

A

*Xwininfo* is a utility for displaying information about windows. Various information is displayed depending on which options are selected. If no options are chosen, **-stats** is assumed.

The user has the option of selecting the target window with the mouse (by clicking any mouse button in the desired window) or by specifying its window id on the command line with the **-id** option. Or instead of specifying the window by its id number, the **-name** option may be used to specify which window is desired by name. There is also a special **-root** option to quickly obtain information on the screen's root window.

## OPTIONS

**-help** Print out the 'Usage:' command syntax summary.

**-id** *id* This option allows the user to specify a target window *id* on the command line rather than using the mouse to select the target window. This is very useful in debugging X applications where the target window is not mapped to the screen or where the use of the mouse might be impossible or interfere with the application.

**-name** *name*
This option allows the user to specify that the window named *name* is the target window on the command line rather than using the mouse to select the target window.

**-root** This option specifies that X's root window is the target window. This is useful in situations where the root window is completely obscured.

**-int** This option specifies that all X window ids should be displayed as integer values. The default is to display them as hexadecimal values.

**-children**
This option causes the root, parent, and children windows' ids and names of the selected window to be displayed.

**-tree** This option is like **-children** but displays all children recursively.

**-stats** This option causes the display of various attributes pertaining to the location and appearance of the selected window. Information displayed includes the location of the window, its width and height, its depth, border width, class, colormap id if any, map state, backing-store hint, and location of the corners.

**-bits** This option causes the display of various attributes pertaining to the selected window's raw bits and how the selected window is to be stored. Displayed information includes the selected window's bit gravity, window gravity, backing-store hint, backing-planes value, backing pixel, and whether or not the window has save-under set.

**-events** This option causes the selected window's event masks to be displayed. Both the event mask of events wanted by some client and the event mask of events not to propagate are displayed.

**-size** This option causes the selected window's sizing hints to be displayed. Displayed information includes: for both the normal size hints and the zoom size hints, the user supplied location if any; the program supplied location if any; the user supplied size if any; the program supplied size if any; the minimum size if any; the maximum size if any; the resize increments if any; and the minimum and maximum aspect ratios if any.

**-wm** This option causes the selected window's window manager hints to be displayed. Information displayed may include whether or not the application accepts input, what the window's icon window # and name is, where the window's icon should go, and what the window's initial state should be.

**-shape** This option causes the selected window's window and border shape extents to be displayed.

**-frame** This option causes window manager frames to be considered when manually selecting windows.

**-metric** This option causes all individual height, width, and x and y positions to be displayed in millimeters as well as number of pixels, based on what the server thinks the resolution is. Geometry specifications that are in +x+y form are not changed.

**-english** This option causes all individual height, width, and x and y positions to be displayed in inches (and feet, yards, and miles if necessary) as well as number of pixels. **-metric** and **-english** may both be enabled at the same time.

**-all** This option is a quick way to ask for all information possible.

**-display** *display*
This option allows you to specify the server to connect to; see *X(1)*.

**EXAMPLE**

The following is a sample summary taken with no options specified:

```
xwininfo: Window id: 0x60000f "xterm"

  Absolute upper-left X: 2
  Absolute upper-left Y: 85
  Relative upper-left X:  0
  Relative upper-left Y:  25
  Width: 579
  Height: 316
  Depth: 8
  Visual Class: PseudoColor
  Border width: 0
  Class: InputOutput
  Colormap: 0x27 (installed)
  Bit Gravity State: NorthWestGravity
  Window Gravity State: NorthWestGravity
  Backing Store State: NotUseful
  Save Under State: no
  Map State: IsViewable
  Override Redirect State: no
  Corners:  +2+85  -699+85  -699-623  +2-623
  -geometry 80x24+0+58
```

**ENVIRONMENT**

**DISPLAY**
To get the default host and display number.

**SEE ALSO**

X(1), xprop(1)

**BUGS**

Using **-stats -bits** shows some redundant information.

The -geometry string displayed must make assumptions about the window's border width and the behavior of the application and the window manager. As a result, the location given is not always correct.

**COPYRIGHT**

**AUTHOR**

Mark Lillibridge, MIT Project Athena

## NAME

xwud - image displayer for X

## SYNOPSIS

**xwud** [-in *file*] [-noclick] [-geometry *geom*] [-display *display*] [-new] [-std <maptype>] [-raw] [-vis <vis-type-or-id>] [-help] [-rv] [-plane *number*] [-fg *color*] [-bg *color*]

## DESCRIPTION

*Xwud* is an X Window System image undumping utility. *Xwud* allows X users to display in a window an image saved in a specially formatted dump file, such as produced by *xwd(1)*.

## OPTIONS

**-bg** *color*
If a bitmap image (or a single plane of an image) is displayed, this option can be used to specify the color to display for the "0" bits in the image.

**-display** *display*
This option allows you to specify the server to connect to; see *X(1)*.

**-fg** *color* If a bitmap image (or a single plane of an image) is displayed, this option can be used to specify the color to display for the "1" bits in the image.

**-geometry** *geom*
This option allows you to specify the size and position of the window. Typically you will only want to specify the position, and let the size default to the actual size of the image.

**-help** Print out a short description of the allowable options.

**-in** *file* This option allows the user to explicitly specify the input file on the command line. If no input file is given, the standard input is assumed.

**-new** This option forces creation of a new colormap for displaying the image. If the image characteristics happen to match those of the display, this can get the image on the screen faster, but at the cost of using a new colormap (which on most displays will cause other windows to go technicolor).

**-noclick** Clicking any button in the window will terminate the application, unless this option is specified. Termination can always be achieved by typing 'q', 'Q', or ctrl-c.

**-plane** *number*
You can select a single bit plane of the image to display with this option. Planes are numbered with zero being the least significant bit. This option can be used to figure out which plane to pass to *xpr(1)* for printing.

**-raw** This option forces the image to be displayed with whatever color values happen to currently exist on the screen. This option is mostly useful when undumping an image back onto the same screen that the image originally came from, while the original windows are still on the screen, and results in getting the image on the screen faster.

**-rv** If a bitmap image (or a single plane of an image) is displayed, this option forces the foreground and background colors to be swapped. This may be needed when displaying a bitmap image which has the color sense of pixel values "0" and "1" reversed from what they are on your display.

**-std** *maptype*
This option causes the image to be displayed using the specified Standard Colormap. The property name is obtained by converting the type to upper case, prepending "RGB_", and appending "_MAP". Typical types are "best", "default", and "gray". See *xstdcmap(1)* for one way of creating Standard Colormaps.

**-vis** *vis-type-or-id*
This option allows you to specify a particular visual or visual class. The default is to pick the "best" one. A particular class can be specified: "StaticGray", "GrayScale", "StaticColor", "PseudoColor", "DirectColor", or "TrueColor". Or "Match" can be specified, meaning use the same class as the source image. Alternatively, an exact visual id (specific to the server) can be specified, either as a hexadecimal number (prefixed with "0x") or as a decimal number. Finally, "default" can be specified, meaning to use the

same class as the colormap of the root window. Case is not significant in any of these strings.

**ENVIRONMENT**

**DISPLAY**
To get default display.

**FILES**

**XWDFile.h**
X Window Dump File format definition file.

**SEE ALSO**

A

xwd(1), xpr(1), X(1)

**COPYRIGHT**

**AUTHOR**

Bob Scheifler, MIT X Consortium

## NAME

bitmap - bitmap editor for the X Window System

## SYNOPSIS

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

## DESCRIPTION

**NOTE:** The supported bitmap editor in HP-UX Release 9.0 is *vueicon. bitmap* is not supported by HP in HP-UX Release 9.0. It is included in the release, but it is now installed in the directory /usr/contrib/bin/X11. Your $PATH environment variable should contain this directory path in order to run this client. See the release notes for more information.

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

## COMMAND LINE OPTIONS

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

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

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

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

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

**-grid, +grid**
Turns on or off the grid lines.

**-axes, +axes**
Turns on or off the major axes.

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

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

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

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

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

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

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

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

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

## USAGE

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

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

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

## EDITING

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

| | |
|---|---|
| MouseButton1 | Set |
| MouseButton2 | Invert |
| MouseButton3 | Clear |
| MouseButton4 | Clear |
| MouseButton5 | Clear |

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

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

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

## DRAWING COMMANDS

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

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

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

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

**Mark**
This command is used to mark an area of the grid by dragging out a rectangular shape in the highlighting color. Once the area is marked, it can be operated on by a number of commands (see **Up, Down, Left, Right, Rotate, Flip, Cut,** etc.) Only one marked area can be present at

any time. If you attempt to mark another area, the old mark will vanish. The same effect can be achieved by pressing **Shift-MouseButton1** and dragging out a rectangle in the grid window. Pressing **Shift-MouseButton2** will mark the entire grid area.

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

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

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

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

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

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

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

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

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

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

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

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

bitmap window has the same effect.

**Point**

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

**Curve**

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

**Line**

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

**Rectangle**

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

**Filled Rectangle**

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

**Circle**

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

**Filled Circle**

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

**Flood Fill**

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

**Set Hot Spot**

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

**Clear Hot Spot**

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

**Undo**

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

## FILE MENU

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

**New**

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

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

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

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

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

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

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

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

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

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

## EDIT MENU

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

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

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

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

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

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

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

A

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

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

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

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

## CUT AND PASTE

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

## WIDGETS

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

```
Bitmap bitmap
        TransientShell image
                Box box
                        Label normalImage
                        Label invertedImage
        TransientShell input
                Dialog dialog
                        Command okay
                        Command cancel
```

```
TransientShell error
        Dialog dialog
                Command abort
                Command retry
TransientShell qsave
        Dialog dialog
                Command yes
                Command no
                Command cancel
Paned parent
        Form formy
                MenuButton fileButton
                SimpleMenu fileMenu
                        SmeBSB  new
                        SmeBSB  load
                        SmeBSB  insert
                        SmeBSB  save
                        SmeBSB  saveAs
                        SmeBSB  resize
                        SmeBSB  rescale
                        SmeBSB  filename
                        SmeBSB  basename
                        SmeLine line
                        SmeBSB  quit
                MenuButton editButton
                SimpleMenu editMenu
                        SmeBSB  image
                        SmeBSB  grid
                        SmeBSB  dashed
                        SmeBSB  axes
                        SmeBSB  stippled
                        SmeBSB  proportional
                        SmeBSB  zoom
                        SmeLine line
                        SmeBSB  cut
                        SmeBSB  copy
                        SmeBSB  paste
                Label status
        Pane pane
                Bitmap bitmap
                Form form
                        Command clear
                        Command set
                        Command invert
                        Toggle  mark
                        Command unmark
                        Toggle  copy
                        Toggle  move
                        Command flipHoriz
                        Command up
                        Command flipVert
                        Command left
                        Command fold
                        Command right
                        Command rotateLeft
                        Command down
                        Command rotateRight
```

A

```
Toggle  point
Toggle  curve
Toggle  line
Toggle  rectangle
Toggle  filledRectangle
Toggle  circle
Toggle  filledCircle
Toggle  floodFill
Toggle  setHotSpot
Command clearHotSpot
Command undo
```

A

**COLORS**

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

```
*customization:          -color
```

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

**BITMAP WIDGET**

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

Bitmap Widget

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

All the Simple Widget resources plus ...

| Name | Class | Type | Default Value |
|---|---|---|---|
| foreground | Foreground | Pixel | XtDefaultForeground |
| highlight | Highlight | Pixel | XtDefaultForeground |
| framing | Framing | Pixel | XtDefaultForeground |
| gridTolerance | GridTolerance | Dimension | 8 |
| size | Size | String | 32x32 |
| dashed | Dashed | Boolean | True |
| grid | Grid | Boolean | True |
| stippled | Stippled | Boolean | True |
| proportional | Proportional | Boolean | True |
| axes | Axes | Boolean | False |
| squareWidth | SquareWidth | Dimension | 16 |
| squareHeight | SquareHeight | Dimension | 16 |
| margin | Margin | Dimension | 16 |
| xHot | XHot | Position | NotSet (-1) |
| yHot | YHot | Position | NotSet (-1) |
| button1Function | Button1Function | DrawingFunction | Set |
| button2Function | Button2Function | DrawingFunction | Invert |
| button3Function | Button3Function | DrawingFunction | Clear |
| button4Function | Button4Function | DrawingFunction | Invert |
| button5Function | Button5Function | DrawingFunction | Invert |

| | | | |
|---|---|---|---|
| filename | Filename | String | None ("") |
| basename | Basename | String | None ("") |

A

## NAME

columns - X11 strategy game

## SYNOPSIS

**columns**

## DESCRIPTION

NOTE: This client is not supported by HP in HP-UX Release 9.0. It is included in the release, but it is now installed in the directory /usr/contrib/bin/X11. Your $PATH environment variable should contain this directory path in order to run this client. See the release notes for more information.

*columns* is a single-player game in which one attempts to manipulate multi-colored tiles in order to form sequences of three boxes of the same color--arranged horizontally, vertically, or diagonally.

The *columns* game window is a 6 x 18 array of squares. The playing tiles drop from the top of the screen, and are comprised of 3 colored boxes arranged in a column. The color of each box in a tile is assigned at random from a pool of--at most--six possible colors: red, green, blue, yellow, magenta, and cyan.

Tiles drop one at a time, and may be manipulated in one of four different ways by 1) moving the tile to the left one position (the *left* command), 2) moving the tile to the right one position (*right*), 3) cycling the colors of the boxes in the tile such that each color moves down one box, with the top box in the tile assuming the color of the bottom box (*cycle*), and 4) dropping the tile straight down into place (*drop*).

The game ends when a tile is placed and--after counting all sequences and removing the boxes involved--a box remains touching the ceiling line near the top of the game window.

## LEVELS OF DIFFICULTY

As mentioned above, the pool of colors from which the boxes in a given tile are assigned is--at most--six. In practice, however, there are twelve different difficulty levels for which the actual number of colors in the pool and the constraints upon them are unique.

At the lowest level, only the first three colors from the pool are used and there are no constraints on how the boxes in a tile are assigned.

At the highest level, the pool includes all six possible colors, and the boxes in the tile are constrained such that none of them may be of the same color.

The complete table depicting the size of the pool and the constraints on the boxes in a tile at each level is as follows:

| LEVEL | # OF COLORS IN THE POOL | MAX # OF BOXES OF ANY ONE COLOR |
|---|---|---|
| 1 | 3 | 3 |
| 2 | " | 2 |
| 3 | " | 1 |
| 4 | 4 | 3 |
| 5 | " | 2 |
| 6 | " | 1 |
| 7 | 5 | 3 |
| 8 | " | 2 |
| 9 | " | 1 |
| 10 | 6 | 3 |
| 11 | " | 2 |
| 12 | " | 1 |

The difficulty level begins at one and increases by one every tenth tile.

**SCORING**

The primary object of the game is to arrange the tiles on the game window such that boxes of the same color line up in three-box sequences. A sequence may be formed horizontally, vertically, or diagonally.

Any time a three-box sequence is produced, the three boxes forming the sequence are removed from the game window, and all boxes located above those removed drop to occupy the vacated squares on the game window. This adds an interesting twist to the game in that new three-box sequences may be produced in a "chain-reaction" fashion as a result of boxes dropping to fill in squares vacated by other boxes. For the purposes of explanation, each phase of sequence-forming is referred to as a "chain".

Each sequence formed scores 10 points if formed during the first chain, 20 points if formed during the second "chain", and so on, with the points-per-sequence increasing by 10 points with each new chain.

Points are also awarded based on the current difficulty level, and on how quickly a tile is dropped down into place by the player.

The points awarded for placing a tile are equivalent to the level number. Thus, six points are awarded for each tile successfully placed when at level six.

The points awarded for placing a tile quickly are equivalent to the number of rows that the tile drops when the DROP keyboard command is issued. Thus, 18 points are awarded if the tile is dropped to the bottom of the game window at the moment it first appears at the top of the game window.

**COMMANDS**

**4, 5, 6, 2**
These are the *left*, *cycle*, *right*, and *drop* commands as mapped onto a numeric keypad.

**J, K, L, <SPACE>**
These are the *left*, *cycle*, *right*, and *drop* keyboard commands for a right-handed player.

**S, D, F, <SPACE>**
These are the *left*, *cycle*, *right*, and *drop* keyboard commands for a left-handed player.

**I** Iconify the game window. ("Yipes! The boss!")

**Q** Quit the current game and exit.

**R** Quit the current game and start over.

All other keys cause the game to pause. Play is resumed by pressing any key besides **Q** or **R**.

**AUTHOR**

Jay Geertsen, Hewlett-Packard Company

## NAME

*dr_vue* - diagnose and report on the HP VUE environment

## SYNOPSIS

**dr_vue**

## DESCRIPTION

NOTE: This client is not supported by HP in HP-UX Release 9.0. It is included in the release, but it is now installed in the directory /usr/contrib/bin/X11. Your $PATH environment variable should contain this directory path in order to run this client. See the release notes for more information.

A

*dr_vue* is a script that checks the integrity of the X Window Systems fileset as it relates to the running of HP VUE. It checks for the existence and permissions of HP VUE files, as well as the status of some other files that HP VUE is dependent upon to run successfully. If there are problems, or information that the user may wish to know, *dr_vue* issues statements informing the user to that effect.

*dr_vue* checks networking to see if the system can ping itself, if the system is properly named, and to see if hosts listed in */etc/X0.hosts* can be reached. If a host cannot be reached, *dr_vue* informs the user of that host's name.

*dr_vue* checks the current run-level of the system, the *initdefault* run-level, and the run-level that HP VUE is set to respawn. If these are different from the default levels that HP ships, *dr_vue* informs the user.

*dr_vue* checks the softbench configuration file, *softinit* (or *$HOME/.softinit*), for the default HP VUE tools that should be listed. If one of them is not listed, *dr_vue* informs the user. *dr_vue* also prints the date that the file was created.

*dr_vue* checks the */usr/adm/inetd.sec* file to insure that the system name is represented in the mserve and spc lines in the file. If they are not literally represented, *dr_vue* will issue a statement requesting that the user insure that the system is represented by domain, wildcard, or inet address.

*dr_vue* checks $HOME to see if a .Xdefaults file exists. If one exists, *dr_vue* informs the user that HP VUE does not use the resources in the file, but instead uses a resource database into which the user can merge defaults. *dr_vue* supplies the user with the proper command to merge the into the database.

*dr_vue* checks the .vueprofile file for stty, tset, or prompt commands. If they exist, *dr_vue* issues a warning. It also checks the current environment for the $DISPLAY variable. If it is set different than <hostname>:0, *dr_vue* informs the user that HP VUE sets the $DISPLAY variable.

*dr_vue* checks the existence and permissions of HP VUE files and directories. If there are missing files or directories, or improper permissions, *dr_vue* reports the problem. *dr_vue* checks for the correct number of links in the special executables that are built together for the VUE environment. If the link count is incorrect, or the "what" string is inaccurate, *dr_vue* informs the user.

## DEPENDENCIES

*dr_vue* must be run on a system that contains the Hewlett Packard Visual User Environment, Version 3.0.

## ENVIRONMENT

*dr_vue* can be run from the command line whether or not X or HP VUE is currently running. *dr_vue* also has an action associated with it, and it can be initiated from the General/Unsupported toolbox if the user is currently running HP VUE Version 3.0.

## ORIGIN

Hewlett-Packard

## NAME

vuefincalc - financial calculator tool

## SYNOPSIS

**/usr/contrib/bin/X11/vuefincalc** [-display *display*] [-geometry *geometry*]

## DESCRIPTION

NOTE: This client is not supported by HP in HP-UX Release 9.0. It is included in the release, but it is now installed in the directory /usr/contrib/bin/X11. Your $PATH environment variable should contain this directory path in order to run this client. See the release notes for more information.

Vuefincalc is a financial calculator tool with a Motif-based user interface. It provides the following capabilities:

A

• RPN (Reverse Polish Notation) calculator with four functions, percent, exponential, logarithmic, square root, integer/fraction, and stack manipulation functions.

• Solve Time Value of Money problems (n, i, PV, PMT, and FV).

• Compute amortization tables.

• Compute depreciation tables (using straight-line, declining balance, Modified ACRS, and sum-of-year-digits depreciation methods).

• Perform discounted cash flow analysis computations (NPV and IRR).

By using large scrolled-text displays instead of a one-line calculator display, Vuefincalc allows you to easily scroll through or cut/paste the calculator stack or entire depreciation or amortization tables.

The Vuefincalc keyboard is divided into two areas, each with its own display:

• The right side is the RPN calculator. It allows you to enter numbers and to perform assorted math and stack functions.

• The left side is the financial calculator, containing the various financial registers and functions. The TVM register keys (n, i, PV, PMT, FV, and others) act as both store and solve keys -- much as on HP financial calculators.

The calculator interface should be reasonably intuitive for users of HP financial calculators. Detailed help in using Vuefincalc is available through the built-in Help facility. When the Help dialog is displayed, you can click on any key, any keyboard label, or any display area for information on how to use those areas.

## OPTIONS

**-display** *display*
: Specifies the server to use. See *X(1)* for more information.

**-geometry** *geometry*
: Specifies the preferred size and position of the vuefincalc window.

## RESOURCES

In addition to the appearance resources (see the file /usr/lib/X11/app-defaults/Vuefincalc), vuefincalc defines the following resources:

**displaySetting (class DisplaySetting)**
: The default display setting for display of numbers. If not specified, it defaults to the locale-dependent value for currency display (2 digits past the radix in most locales).

**stackSize (class StackSize)**
: The size of the math stack in the RPN calculator. Defaults to an infinite stack.

**enterVertical (class EnterVertical)**
: If True, the lettering on the "ENTER" key is vertical instead of horizontal. With a proportionally-spaced font, this can result in reducing the amount of screen space taken by vuefincalc. Defaults to false.

**RELATED FILES**

*/usr/lib/X11/app-defaults/Vuefincalc*
*/usr/lib/nls/C/vuefincalc.cat*

**ORIGIN**

HP

A

## NAME

xclipboard - X clipboard client

## SYNOPSIS

**xclipboard** [ *-toolkitoption* ...] [-w] [-nw]

## DESCRIPTION

NOTE: This client is not supported by HP in HP-UX Release 9.0. It is included in the release, but it is now installed in the directory /usr/contrib/bin/X11. Your $PATH environment variable should contain this directory path in order to run this client. See the release notes for more information.

A

The *xclipboard* program is used to collect and display text selections that are sent to the CLIPBOARD by other clients. It is typically used to save CLIPBOARD selections for later use. It stores each CLIPBOARD selection as a separate string, each of which can be selected. Each time CLIPBOARD is asserted by another application, *xclipboard* transfers the contents of that selection to a new buffer and displays it in the text window. Buffers are never automatically deleted, so you'll want to use the delete button to get rid of useless items.

Since *xclipboard* uses a Text Widget to display the contents of the clipboard, text sent to the CLIPBOARD may be re-selected for use in other applications. *xclipboard* also responds to requests for the CLIPBOARD selection from other clients by sending the entire contents of the currently displayed buffer.

An *xclipboard* window has the following buttons across the top:

| | |
|---|---|
| *quit* | When this button is pressed, *xclipboard* exits. |
| *delete* | When this button is pressed, the current buffer is deleted and the next one displayed. |
| *new* | Creates a new buffer with no contents. Useful in constructing a new CLIPBOARD selection by hand. |
| *next* | Displays the next buffer in the list. |
| *previous* | Displays the previous buffer. |

## OPTIONS

The *xclipboard* program accepts all of the standard X Toolkit command line options as well as the following:

| | |
|---|---|
| **-w** | This option indicates that lines of text that are too long to be displayed on one line in the clipboard should wrap around to the following lines. |
| **-nw** | This option indicates that long lines of text should not wrap around. This is the default behavior. |

## WIDGETS

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

```
XClipboard  xclipboard
        Form  form
                Command  quit
                Command  delete
                Command  new
                Command  next
                Command  prev
                Text  text
```

## SENDING/RETRIEVING CLIPBOARD CONTENTS

Text is copied to the clipboard whenever a client asserts ownership of the **CLIPBOARD** selection. Text is copied from the clipboard whenever a client requests the contents of the **CLIPBOARD** selection. Examples of event bindings that a user may wish to include in a resource configuration file to use the clipboard are:

```
*VT100.Translations: #override \
        <Btn3Up>:                       select-end(CLIPBOARD) \n\
        <Btn2Up>:                       insert-selection(PRIMARY,CLIPBOARD) \
        <Btn2Down>:                     ignore ()
```

**SEE ALSO**

X(1), xcutsel(1), xterm(1), individual client documentation for how to make a selection and send it to the CLIPBOARD.

A

**ENVIRONMENT**

**DISPLAY**
to get the default host and display number.

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

**FILES**

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

**COPYRIGHT**

Copyright 1988, Massachusetts Institute of Technology
See *X(1)* for a full statement of rights and permissions.

**AUTHOR**

Ralph R. Swick, DEC/MIT Project Athena
Chris D. Peterson, MIT X Consortium
Keith Packard, MIT X Consortium

## NAME

xcutsel - interchange between cut buffer and selection

## SYNOPSIS

**xcutsel** [ *-toolkitoption* ...] [-selection *selection*] [-cutbuffer *number*]

## DESCRIPTION

NOTE: This client is not supported by HP in HP-UX Release 9.0. It is included in the release, but it is now installed in the directory /usr/contrib/bin/X11. Your $PATH environment variable should contain this directory path in order to run this client. See the release notes for more information.

The *xcutsel* program is used to copy the current selection into a cut buffer and to make a selection that contains the current contents of the cut buffer. It acts as a bridge between applications that don't support selections and those that do.

By default, *xcutsel* will use the selection named PRIMARY and the cut buffer CUT_BUFFER0. Either or both of these can be overridden by command line arguments or by resources.

An *xcutsel* window has the following buttons:

*quit* When this button is pressed, *xcutsel* exits. Any selections held by *xcutsel* are automatically released.

*copy PRIMARY to 0*
When this button is pressed, *xcutsel* copies the current selection into the cut buffer.

*copy 0 to PRIMARY*
When this button is pressed, *xcutsel* converts the current contents of the cut buffer into the selection.

The button labels reflect the selection and cutbuffer selected by command line options or through the resource database.

When the "copy 0 to PRIMARY" button is activated, the button will remain inverted as long as *xcutsel* remains the owner of the selection. This serves to remind you which client owns the current selection. Note that the value of the selection remains constant; if the cutbuffer is changed, you must again activate the copy button to retrieve the new value when desired.

## OPTIONS

*Xcutsel* accepts all of the standard X Toolkit command line options as well as the following:

**-selection** *name*
This option specifies the name of the selection to use. The default is PRIMARY. The only supported abbreviations for this option are "-select", "-sel" and "-s", as the standard toolkit option "-selectionTimeout" has a similar name.

**-cutbuffer** *number*
This option specifies the cut buffer to use. The default is cut buffer 0.

## X DEFAULTS

This program accepts all of the standard X Toolkit resource names and classes as well as:

**selection** (class **Selection**)
This resource specifies the name of the selection to use. The default is PRIMARY.

**cutBuffer** (class **CutBuffer**)
This resource specifies the number of the cut buffer to use. The default is 0.

## WIDGET NAMES

The following instance names may be used when user configuration of the labels in them is desired:

**sel-cut** (class **Command**)
This is the "copy SELECTION to BUFFER" button.

**cut-sel** (class **Command**)
This is the "copy BUFFER to SELECTION" button.

**quit** (class **Command)**
This is the "quit" button.

**SEE ALSO**

X(1), xclipboard(1), xterm(1), text widget documentation, individual client documentation for how to make a selection.

**BUGS**

There is no way to change the name of the selection or the number of the cut buffer while the program is running.

A

**COPYRIGHT**

Copyright 1988, Massachusetts Institute of Technology
See *X(1)* for a full statement of rights and permissions.

**AUTHOR**

Ralph R. Swick, DEC/MIT Project Athena

## NAME

xdpyinfo - display information utility for X

## SYNOPSIS

**xdpyinfo** [-display *displayname*]

## DESCRIPTION

NOTE: This client is not supported by HP in HP-UX Release 9.0. It is included in the release, but it is now installed in the directory /usr/contrib/bin/X11. Your $PATH environment variable should contain this directory path in order to run this client. See the release notes for more information.

*Xdpyinfo* is a utility for displaying information about an X server. It is used to examine the capabilities of a server, the predefined values for various parameters used in communicating between clients and the server, and the different types of screens and visuals that are available.

A

## EXAMPLE

The following shows a sample produced by *xdpyinfo* when connected to display that supports an 8 plane screen and a 1 plane screen.

```
name of display:   :0.0
version number:   11.0
vendor string:   MIT X Consortium
vendor release number:   4
maximum request size:  16384 longwords (65536 bytes)
motion buffer size:  0
bitmap unit, bit order, padding:   32, MSBFirst, 32
image byte order:   MSBFirst
number of supported pixmap formats:   2
supported pixmap formats:
   depth 1, bits_per_pixel 1, scanline_pad 32
   depth 8, bits_per_pixel 8, scanline_pad 32
keycode range:   minimum 8, maximum 129
focus:  PointerRoot
number of extensions:   4
   SHAPE
   MIT-SHM
   Multi-Buffering
   MIT-SUNDRY-NONSTANDARD
default screen number:   0
number of screens:   2

screen #0:
  dimensions:   1152x900 pixels (325x254 millimeters)
  resolution:   90x90 dots per inch
  depths (2):   1, 8
  root window id:   0x8006e
  depth of root window:   8 planes
  number of colormaps:   minimum 1, maximum 1
  default colormap:   0x8006b
  default number of colormap cells:   256
  preallocated pixels:   black 1, white 0
  options:   backing-store YES, save-unders YES
  current input event mask:   0xd0801d
    KeyPressMask          ButtonPressMask        ButtonReleaseMask
    EnterWindowMask        ExposureMask          SubstructureRedirectMask
    PropertyChangeMask      ColormapChangeMask
  number of visuals:   6
  default visual id:  0x80065
  visual:
    visual id:   0x80065
    class:    PseudoColor
    depth:    8 planes
    available colormap entries:    256
    red, green, blue masks:    0x0, 0x0, 0x0
    significant bits in color specification:    8 bits
  visual:
    visual id:    0x80066
    class:    DirectColor
    depth:    8 planes
    available colormap entries:    8 per subfield
    red, green, blue masks:    0x7, 0x38, 0xc0
    significant bits in color specification:    8 bits
  visual:
    visual id:    0x80067
    class:    GrayScale
    depth:    8 planes
    available colormap entries:    256
    red, green, blue masks:    0x0, 0x0, 0x0
    significant bits in color specification:    8 bits
  visual:
    visual id:    0x80068
    class:    StaticGray
    depth:    8 planes
    available colormap entries:    256
    red, green, blue masks:    0x0, 0x0, 0x0
    significant bits in color specification:    8 bits
  visual:
    visual id:    0x80069
    class:    StaticColor
    depth:    8 planes
    available colormap entries:    256
    red, green, blue masks:    0x7, 0x38, 0xc0
    significant bits in color specification:    8 bits
  visual:
    visual id:    0x8006a
    class:    TrueColor
    depth:    8 planes
    available colormap entries:    8 per subfield
    red, green, blue masks:    0x7, 0x38, 0xc0
    significant bits in color specification:    8 bits
  number of mono multibuffer types:    6
    visual id, max buffers, depth:    0x80065, 0, 8
    visual id, max buffers, depth:    0x80066, 0, 8
    visual id, max buffers, depth:    0x80067, 0, 8
    visual id, max buffers, depth:    0x80068, 0, 8
    visual id, max buffers, depth:    0x80069, 0, 8
    visual id, max buffers, depth:    0x8006a, 0, 8
  number of stereo multibuffer types:    0

screen #1:
  dimensions:    1152x900 pixels (325x254 millimeters)
  resolution:    90x90 dots per inch
  depths (1):    1
  root window id:    0x80070
  depth of root window:    1 plane
  number of colormaps:    minimum 1, maximum 1
  default colormap:    0x8006c
```

```
    default number of colormap cells:    2
    preallocated pixels:    black 1, white 0
    options:    backing-store YES, save-unders YES
    current input event mask:    0xd0801d
      KeyPressMask            ButtonPressMask         ButtonReleaseMask
      EnterWindowMask         ExposureMask            SubstructureRedirectMask
      PropertyChangeMask      ColormapChangeMask
    number of visuals:    1
    default visual id:  0x80064
    visual:
      visual id:    0x80064
      class:    StaticGray
      depth:    1 plane
      available colormap entries:    2
      red, green, blue masks:    0x0, 0x0, 0x0
      significant bits in color specification:    1 bits
    number of mono multibuffer types:    1
      visual id, max buffers, depth:    0x80064, 0, 1
    number of stereo multibuffer types:    0
```

A

## ENVIRONMENT

DISPLAY
: To get the default host, display number, and screen.

## SEE ALSO

X(1), xwininfo(1), xprop(1), xrdb(1)

## COPYRIGHT

Copyright 1988, 1989, Massachusetts Institute of Technology.
See *X(1)* for a full statement of rights and permissions.

## AUTHOR

Jim Fulton, MIT X Consortium

## NAME

xfd - display all the characters in an X font

## SYNOPSIS

**xfd** [-options ...] -fn *fontname*

## DESCRIPTION

**NOTE:** This client is not supported by HP in HP-UX Release 9.0. It is included in the release, but it is now installed in the directory /usr/contrib/bin/X11. Your $PATH environment variable should contain this directory path in order to run this client. See the release notes for more information.

A

The *xfd* utility creates a window containing the name of the font being displayed, a row of command buttons, several lines of text for displaying character metrics, and a grid containing one glyph per cell. The characters are shown in increasing order from left to right, top to bottom. The first character displayed at the top left will be character number 0 unless the **-start** option has been supplied in which case the character with the number given in the **-start** option will be used.

The characters are displayed in a grid of boxes, each large enough to hold any single character in the font. Each character glyph is drawn using the PolyText16 request (used by the *Xlib* routine **XDrawString16**). If the **-box** option is given, a rectangle will be drawn around each character, showing where an ImageText16 request (used by the *Xlib* routine **XDrawImageString16**) would cause background color to be displayed.

The origin of each glyph is normally set so that the character is drawn in the upper left hand corner of the grid cell. However, if a glyph has a negative left bearing or an unusually large ascent, descent, or right bearing (as is the case with *cursor* font), some character may not appear in their own grid cells. The **-center** option may be used to force all glyphs to be centered in their respective cells.

All the characters in the font may not fit in the window at once. To see the next page of glyphs, press the *Next* button at the top of the window. To see the previous page, press *Prev*. To exit *xfd*, press *Quit*.

Individual character metrics (index, width, bearings, ascent and descent) can be displayed at the top of the window by pressing on the desired character.

The font name displayed at the top of the window is the full name of the font, as determined by the server. See *xlsfonts* for ways to generate lists of fonts, as well as more detailed summaries of their metrics and properties.

## OPTIONS

*xfd* accepts all of the standard toolkit command line options along with the additional options listed below:

**-fn** *font* This option specifies the font to be displayed.

**-box** This option indicates that a box should be displayed outlining the area that would be filled with background color by an ImageText request.

**-center** This option indicates that each glyph should be centered in its grid.

**-start** *number*
This option specifies the glyph index of the upper left hand corner of the grid. This is used to view characters at arbitrary locations in the font. The default is 0.

**-bc** *color*
This option specifies the color to be used if ImageText boxes are drawn.

## X DEFAULTS

To be written.

## SEE ALSO

X(1), xlsfonts(1), xrdb(1)

## BUGS

The program should skip over pages full of non-existent characters.

**COPYRIGHT**

**AUTHOR**

Jim Fulton, MIT X Consortium; previous program of the same name by Mark Lillibridge, MIT Project Athena.

## NAME

xprop - property displayer for X

## SYNOPSIS

**xprop** [-help] [-grammar] [-id *id*] [-root] [-name *name*] [-frame] [-font *font*] [-display *display*] [-len *n*] [-notype] [-fs *file*] [-remove *property-name*] [-spy] [-f *atom format* [*dformat*]]* [*format* [*dformat*] *atom*]*

## SUMMARY

A

NOTE: This client is not supported by HP in HP-UX Release 9.0. It is included in the release, but it is now installed in the directory /usr/contrib/bin/X11. Your $PATH environment variable should contain this directory path in order to run this client. See the release notes for more information.

The *prop* utility is for displaying window and font properties in an X server. One window or font is selected using the command line arguments or possibly in the case of a window, by clicking on the desired window. A list of properties is then given, possibly with formatting information.

## OPTIONS

**-help** Print out a summary of command line options.

**-grammar**
Print out a detailed grammar for all command line options.

**-id** *id* This argument allows the user to select window *id* on the command line rather than using the pointer to select the target window. This is very useful in debugging X applications where the target window is not mapped to the screen or where the use of the pointer might be impossible or interfere with the application.

**-name** *name*
This argument allows the user to specify that the window named *name* is the target window on the command line rather than using the pointer to select the target window.

**-font** *font*
This argument allows the user to specify that the properties of font *font* should be displayed.

**-root** This argument specifies that X's root window is the target window. This is useful in situations where the root window is completely obscured.

**-display** *display*
This argument allows you to specify the server to connect to; see *X(1)*.

**-len** *n* Specifies that at most *n* bytes of any property should be read or displayed.

**-notype** Specifies that the type of each property should not be displayed.

**-fs** *file* Specifies that file *file* should be used as a source of more formats for properties.

**-frame** Specifies that when selecting a window by hand (i.e. if none of **-name**, **-root**, or **-id** are given), look at the window manager frame (if any) instead of looking for the client window.

**-remove** *property-name*
Specifies the name of a property to be removed from the indicated window.

**-spy** Examine window properties forever, looking for property change events.

**-f** *name format* [*dformat*]
Specifies that the *format* for *name* should be *format* and that the *dformat* for *name* should be *dformat*. If *dformat* is missing, " = $0+\n" is assumed.

## DESCRIPTION

For each of these properties, its value on the selected window or font is printed using the supplied formatting information if any. If no formatting information is supplied, internal defaults are used. If a property is not defined on the selected window or font, "not defined" is printed as the value for that property. If no property list is given, all the properties possessed by the selected window or font are printed.

A window may be selected in one of four ways. First, if the desired window is the root window, the -root argument may be used. If the desired window is not the root window, it may be selected in two ways on the command line, either by id number such as might be obtained from *xwininfo*, or by name if the window possesses a name. The -id argument selects a window by id number in either decimal or hex (must start with 0x) while the -name argument selects a window by name.

The last way to select a window does not involve the command line at all. If none of -font, -id, -name, and -root are specified, a crosshairs cursor is displayed and the user is allowed to choose any visible window by pressing any pointer button in the desired window. If it is desired to display properties of a font as opposed to a window, the -font argument must be used.

Other than the above four arguments and the -help argument for obtaining help, and the -grammar argument for listing the full grammar for the command line, all the other command line arguments are used in specifying both the format of the properties to be displayed and how to display them. The -len *n* argument specifies that at most *n* bytes of any given property will be read and displayed. This is useful for example when displaying the cut buffer on the root window which could run to several pages if displayed in full.

Normally each property name is displayed by printing first the property name then its type (if it has one) in parentheses followed by its value. The -notype argument specifies that property types should not be displayed. The -fs argument is used to specify a file containing a list of formats for properties while the -f argument is used to specify the format for one property.

The formatting information for a property actually consists of two parts, a *format* and a *dformat*. The *format* specifies the actual formatting of the property (i.e., is it made up of words, bytes, or longs?, etc.) while the *dformat* specifies how the property should be displayed.

The following paragraphs describe how to construct *formats* and *dformats*. However, for the vast majority of users and uses, this should not be necessary as the built in defaults contain the *formats* and *dformats* necessary to display all the standard properties. It should only be necessary to specify *formats* and *dformats* if a new property is being dealt with or the user dislikes the standard display format. New users especially are encouraged to skip this part.

A *format* consists of one of 0, 8, 16, or 32 followed by a sequence of one or more format characters. The 0, 8, 16, or 32 specifies how many bits per field there are in the property. Zero is a special case meaning use the field size information associated with the property itself. (This is only needed for special cases like type INTEGER which is actually three different types depending on the size of the fields of the property)

A value of 8 means that the property is a sequence of bytes while a value of 16 would mean that the property is a sequence of words. The difference between these two lies in the fact that the sequence of words will be byte swapped while the sequence of bytes will not be when read by a machine of the opposite byte order of the machine that originally wrote the property. For more information on how properties are formatted and stored, consult the Xlib manual.

Once the size of the fields has been specified, it is necessary to specify the type of each field (i.e., is it an integer, a string, an atom, or what?) This is done using one format character per field. If there are more fields in the property than format characters supplied, the last character will be repeated as many times as necessary for the extra fields. The format characters and their meaning are as follows:

| | |
|---|---|
| a | The field holds an atom number. A field of this type should be of size 32. |
| b | The field is an boolean. A 0 means false while anything else means true. |
| c | The field is an unsigned number, a cardinal. |
| i | The field is a signed integer. |
| m | The field is a set of bit flags, 1 meaning on. |
| s | This field and the next ones until either a 0 or the end of the property represent a sequence of bytes. This format character is only usable with a field size of 8 and is most often used to represent a string. |
| x | The field is a hex number (like 'c' but displayed in hex - most useful for displaying window ids and the like) |

An example *format* is 32ica which is the format for a property of three fields of 32 bits each, the first holding a signed integer, the second an unsigned integer, and the third an atom.

The format of a *dformat* unlike that of a *format* is not so rigid. The only limitations on a *dformat* is that one may not start with a letter or a dash. This is so that it can be distinguished from a property name or an argument. A *dformat* is a text string containing special characters instructing that various fields be printed at various points in a manner similar to the formatting string used by printf. For example, the *dformat* " is ( $0, $1 \)\n" would render the POINT 3, -4 which has a *format* of 32ii as " is ( 3, -4 )\n".

Any character other than a $, ?, \, or a ( in a *dformat* prints as itself. To print out one of $, ?, \, or ( precede it by a \. For example, to print out a $, use \$. Several special backslash sequences are provided as shortcuts. \n will cause a newline to be displayed while \t will cause a tab to be displayed. *o* where *o* is an octal number will display character number *o*.

A $ followed by a number *n* causes field number *n* to be displayed. The format of the displayed field depends on the formatting character used to describe it in the corresponding *format*. I.e., if a cardinal is described by 'c' it will print in decimal while if it is described by a 'x' it is displayed in hex.

If the field is not present in the property (this is possible with some properties), <field not available> is displayed instead. $*n*+ will display field number *n* then a comma then field number *n*+1 then another comma then ... until the last field defined. If field *n* is not defined, nothing is displayed. This is useful for a property that is a list of values.

A ? is used to start a conditional expression, a kind of if-then statement. ?*exp*(*text*) will display *text* if and only if *exp* evaluates to non-zero. This is useful for two things. First, it allows fields to be displayed if and only if a flag is set. And second, it allows a value such as a state number to be displayed as a name rather than as just a number. The syntax of *exp* is as follows:

*exp* ::= *term* | *term*=*exp* | !*exp*

*term* ::= *n* | $*n* | m*n*

The ! operator is a logical "not", changing 0 to 1 and any non-zero value to 0. = is an equality operator. Note that internally all expressions are evaluated as 32 bit numbers so -1 is not equal to 65535. = returns 1 if the two values are equal and 0 if not. *n* represents the constant value *n* while $*n* represents the value of field number *n*. m*n* is 1 if flag number *n* in the first field having format character 'm' in the corresponding *format* is 1, 0 otherwise.

Examples: ?m3(count: $3\n) displays field 3 with a label of count if and only if flag number 3 (count starts at 0!) is on. ?$2=0(True)?!$2=0(False) displays the inverted value of field 2 as a boolean.

In order to display a property, *xprop* needs both a *format* and a *dformat*. Before *xprop* uses its default values of a *format* of 32x and a *dformat* of " = { $0+ }\n", it searches several places in an attempt to find more specific formats. First, a search is made using the name of the property. If this fails, a search is made using the type of the property. This allows type STRING to be defined with one set of formats while allowing property WM_NAME which is of type STRING to be defined with a different format. In this way, the display formats for a given type can be overridden for specific properties.

The locations searched are in order: the format if any specified with the property name (as in 8x WM_NAME), the formats defined by -f options in last to first order, the contents of the file specified by the -fs option if any, the contents of the file specified by the environmental variable XPROPFORMATS if any, and finally *xprop*'s built in file of formats.

The format of the files referred to by the -fs argument and the XPROPFORMATS variable is one or more lines of the following form:

*name format* [*dformat*]

Where *name* is either the name of a property or the name of a type, *format* is the *format* to be used with *name* and *dformat* is the *dformat* to be used with *name*. If *dformat* is not present, " = $0+\n" is assumed.

**EXAMPLES**

To display the name of the root window: *xprop* -root WM_NAME

To display the window manager hints for the clock: *xprop* -name xclock WM_HINTS

To display the start of the cut buffer: *xprop* -root -len 100 CUT_BUFFER0

To display the point size of the fixed font: *xprop* -font fixed POINT_SIZE

To display all the properties of window # 0x200007: *xprop* -id 0x200007

**ENVIRONMENT**

**DISPLAY**
To get default display.

**XPROPFORMATS**
Specifies the name of a file from which additional formats are to be obtained.

**SEE ALSO**

X(1), xwininfo(1)

**COPYRIGHT**

**AUTHOR**

Mark Lillibridge, MIT Project Athena

## NAME

xrefresh - refresh all or part of an X screen

## SYNOPSIS

**xrefresh** [-option ...]

## DESCRIPTION

NOTE: This client is not supported by HP in HP-UX Release 9.0. It is included in the release, but it is now installed in the directory /usr/contrib/bin/X11. Your $PATH environment variable should contain this directory path in order to run this client. See the release notes for more information.

A

*Xrefresh* is a simple X program that causes all or part of your screen to be repainted. This is useful when system messages have messed up your screen. *Xrefresh* maps a window on top of the desired area of the screen and then immediately unmaps it, causing refresh events to be sent to all applications. By default, a window with no background is used, causing all applications to repaint "smoothly." However, the various options can be used to indicate that a solid background (of any color) or the root window background should be used instead.

## ARGUMENTS

**-white** Use a white background. The screen just appears to flash quickly, and then repaint.

**-black** Use a black background (in effect, turning off all of the electron guns to the tube). This can be somewhat disorienting as everything goes black for a moment.

**-solid** *color*
Use a solid background of the specified color. Try green.

**-root** Use the root window background.

**-none** This is the default. All of the windows simply repaint.

**-geometry** *WxH+X+Y*
Specifies the portion of the screen to be repainted; see *X(1)*.

**-display** *display*
This argument allows you to specify the server and screen to refresh; see *X(1)*.

## X DEFAULTS

The *xrefresh* program uses the routine *XGetDefault(3X)* to read defaults, so its resource names are all capitalized.

**Black, White, Solid, None, Root**
Determines what sort of window background to use.

**Geometry**
Determines the area to refresh. Not very useful.

## ENVIRONMENT

DISPLAY - To get default host and display number.

## SEE ALSO

X(1)

## BUGS

It should have just one default type for the background.

## COPYRIGHT

## AUTHORS

Jim Gettys, Digital Equipment Corp., MIT Project Athena

B

# Using the Keyboards

There are now two keyboards available for Hewlett-Packard workstations. Until now, the 46021 keyboard, also known as the "ITF" keyboard, was the only keyboard available. Now, in addition to the 46021 keyboard, a personal computer-style keyboard, C1429 is also available. This new keyboard is also known as the "Enhanced Vectra" keyboard.

## Understanding the Keyboards

If an application is reading input directly from the keyboard, it receives a *keycode* when a key is pressed. Equivalent keys on the two keyboards are those that generate the same keycode. If an equivalent key does not exist, there is no way to generate the corresponding keycode.

In an X Window System environment, keycodes are mapped into *key symbols* by the X library. The key symbols are stored in a *keysym table*. Application programs then reference these key symbols when accessing keys.

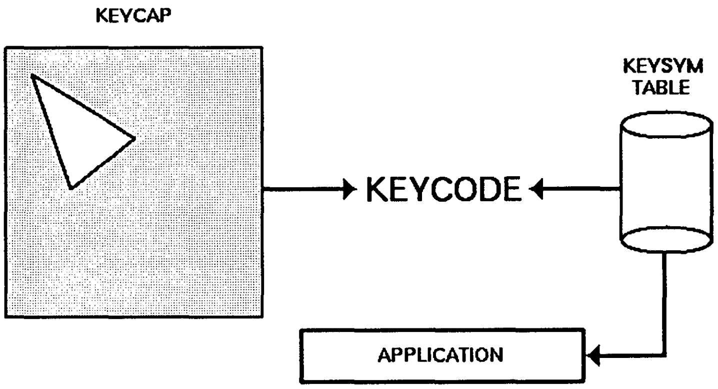


**Figure B-1. Keycap, Keycode, and Keysym Relationships**

Equivalent keys are those keys that are mapped to the same key symbol. One advantage of this mapping is that if a key does not physically exist on a keyboard, its equivalent key symbol can be mapped to some other key through the corresponding keycode.

## Default Keyboard Mapping

The default keyboard mapping supplied with the X Window environment maps the C1429 keyboard to the same key symbols that are used for the 46021 keyboard. This allows existing X client programs that expect to receive input from a 46021 keyboard to be used with either keyboard. However, the result is that some keys on the C1429 keyboard are mapped to key symbols that do not match the engravings on their keycaps.

## Equivalent Keys

Some applications may expect to use keys that exist on one of the keyboards but not the other. In most cases, if a key does not exist on the keyboard in use, it is still possible to use some other key that is equivalent. To do this, it is necessary to know which keys are equivalent on the two keyboards.

There are 14 keys on the C1429 keyboard that generate keycodes equivalent to keys on the 46021 keyboard, but have different engravings on the keycaps. Some have the same key symbol on both keyboards, while others do not. These C1429 keys, their 46021 equivalents, and the corresponding symbol names are shown in the following table.

B

| C1429 Keycap | 46021 Keycap | Default Key Symbol | XPCmodmap Symbol |
|---|---|---|---|
| F9 | blank1 | F9 | F9 |
| F10 | blank2 | F10 | F10 |
| F11 | blank3 | F11 | F11 |
| F12 | blank4 | F12 | F12 |
| PrintScreen/sysRq | Menu | Menu | Print |
| Scroll Lock | Stop | Cancel | Scroll_Lock |
| Pause/Break | Break/Reset | Break/Reset | Pause/Break |
| Page Up | Prev | Prior | Prior |
| Num Lock | System/User | System/User | Num_Lock |
| End | Select | Select | End |
| Page Down | Next | Next | Next |
| Enter | Return | Return | Return |
| Alt (left) | Extend char (left) | Meta_L | Alt_L |
| Alt (right) | Extend char (right) | Meta_R | Alt_R |

## Changing Key Mapping

X provides the means to change the key mapping, if you so desire. One way to accomplish this is by running the `xmodmap` client program. Hewlett-Packard provides two files in the directory `/usr/lib/X11` to use with `xmodmap`. One, `XPCmodmap`, causes `xmodmap` to change the key mapping to match the keycap engravings on the C1429 keyboard. The other, `XHPmodmap`, causes `xmodmap` to change the key mapping to match the keycap engravings on the 46021 keyboard, which are the defaults. This allows either keyboard to be used with applications that expect the other keyboard, although only one mapping can be used at any given time. When the mapping is changed, the X Server notifies all clients that are executing at that time. Some clients may load the new mapping from the server right away, but others may have to be restarted in order to recognize the new mapping. For more information about using the `xmodmap` client, see the `xmodmap` man page. Additional information can be found in chapter 9, Customizing the Mouse and Keyboard, in *Using the X Window System.*

B

### C1429 Keyboard

Execute the following command to change the mapping of the keys shown above to match the engravings on the C1429 keycaps.

```
/usr/bin/X11/xmodmap /usr/lib/X11/XPCmodmap
```

### 46021 Keyboard

Execute the following command to change the mapping to match the 46021 keyboard.

```
/usr/bin/X11/xmodmap /usr/lib/X11/XHPmodmap
```

### Comparing the Keyboards

The 46021 keyboard has 107 keys, while the C1429 keyboard has 101 keys. There are 7 keys on the 46021 keyboard whose keycodes cannot be generated by any key on the C1429 keyboard, and whose key symbols cannot be generated when using the default keymap for the C1429 keyboard. The missing keys are:

- (Clear line)
- (Clear display)

- Insert line
- Delete line
- Print/Enter
- , (on number pad)
- Tab (on number pad)

, and Tab exist elsewhere on the C1429 keyboard, and the others are not needed by most applications. Applications that do need one or more of them must assign their key symbols to the keycodes of existing keys. The `xmodmap` client can be used to determine the keycode-to-key symbol mapping of existing keys, and it can also be used to assign the key symbol to the desired keycode. These keys use HP specific key symbol names whose correct spelling can be found in the file `/usr/lib/X11/XKeysymDB`.

The Right Control key on the C1429 keyboard generates a keycode that has no equivalent on the 46021 keyboard. This key has the same effect as the Left Control key by default.

Keys not mentioned above exist on both keyboards, and have the same key symbols.

# Glossary

**Accelerator**
A key or sequence of keys (typically a modifier key and some other key) that provides a "shortcut," for accessing functionality.

**active window**
The terminal window where what you type appears. If there is no active window, what you type is lost. Only one terminal window can be active at a time.

**application program**
A computer program that performs some useful function, such as word processing or data base management.

**application server**
A computer used solely to provide processing power for application programs.

**ampersand (&)**
Placed at the end of a command to specify that the client started by the command should be started as a background process. The command can be typed after the command-line prompt or included in a file such as `.x11start` or `.hpwmrc`.

**background process**
A process that doesn't require the total attention of the computer for operation. Background processing enables the operating system to execute more than one program or command at a time. As a general rule, all clients should be run as background processes.

**bitmap**
Generally speaking, an array of data bits used for graphic images. Strictly speaking, a pixmap of depth one (capable of 2-color images).

**bitmap device**
An output device that displays bitmaps. The CRT monitor of your system is a bitmap device.

**bitmap font**
A bitmap font is made from a matrix of dots.

**buffer**
An area used for storage.

**button**
A button on a mouse pointing device. Mouse buttons can be mapped to the keyboard.

**button binding**
Association of a mouse button operation with a window manager function. For example, pressing button 3 on a window frame displays the system menu.

**button mapping**
Association of a button number with a physical mouse button.

**click**
To press *and release* a mouse button. The term comes from the fact that pressing and releasing the buttons of most mice makes a clicking sound.

**client**
A program written specifically for the X Window System. Some clients make their own windows. Other clients are utility programs.

**cluster**
A network of computers in which only one computer has file-system disk drives attached to it.

**combined mode**
A combination of image and overlay planes in which a single display has a single screen that is a combination of the image and overlay planes.

**command-line prompt**
A command-line prompt shows that the computer is ready to accept your commands. Each terminal emulation window has a command-line prompt that acts just like the command-line prompt you see on the screen immediately after login. Usually the command-line prompt is either a **$** (for Bourne and Korn shells) or a **%** (for C shells), but it can be modified.

One popular modification is to print the current working directory and the history stack number before the $ or %. You can find the command-line prompt by pressing Return several times. Every time you press Return, HP-UX prints the prompt.

**cut buffer**
A buffer (memory area) that holds text that has been deleted from a file.

**depth**
The number of planes in a set of planes. For example, a set of 12 image planes would have a depth of 12.

**diskless cluster**
The networking of several systems (SPUs) together to share a common hard disk for storage of data and programs.

**display**
Strictly speaking, the combination of a keyboard, mouse, and one or more screens that provide input and output services to a system. While "display" is sometimes used to mean just the CRT screen, a display, as defined here, can actually include more than one physical screen.

**display server**
In the X Window System, the display server is the software that controls the communication between client programs and the display (keyboard, mouse, and screen combination).

**double buffering**
A term describing the method used by Starbase wherein half of the color planes on a monitor are used to display to the screen and the other half are used to compute and draw the next screen display. This provides smooth motion for animation and it is faster. However, it does reduce the number of colors that are available for display on the screen at one time.

**double-click**
To press *and release* a mouse button twice in rapid succession.

**drag**
To press *and hold down* a mouse button while moving the mouse on the desktop (and the pointer on the screen). Typically, dragging is used with menu selecting, moving, and resizing operations.

**file server**
A computer whose primary task is to control the storage and retrieval of data from hard disks. Any number of other computers can be linked to the file server in order to use it to access data. This means that less storage space is required on the individual computer.

**fonts**
A font is a style of printed text characters. Times Roman is the font used for most newspaper text; Helvetica is the font used for most newspaper headlines.

**foreground process**
A process that has the terminal window's attention. When a program is run in a window as a foreground process (as opposed to a background process), the terminal window cannot be used for other operations until the process is terminated.

**graphical user interface**
A form of communication between people and computers that uses graphics-oriented software such as windows, menus, and icons, to ease the burden of the interaction.

**home directory**
The directory in which you are placed after you log in. Typically, this is `/users/`*username*, where *username* is your login name. The home directory is where you keep all "your" files.

**hotspot**
The area of a graphical image used as a pointer or cursor that is defined as the "point" of the pointer or cursor.

**hpterm**
A type of terminal window, sometimes called a "terminal emulator program" that emulates HP2622 terminals, complete with softkeys. The `hpterm` window is the default window for your X environment.

**icon**
A small, graphic representation of an object on the root window (typically a terminal window). Objects can be "iconified" (turned into icons) to clear a cluttered workspace and "normalized" (returned to their original appearance) as needed. Processes executing in an object continue to execute when the object is iconified.

**iconify**
The act of turning a window into an icon.

**image mode**
The default screen mode using multiple image planes for a single screen. The number of image planes determines the variety of colors that are available to the screen.

**image planes**
The primary display planes on a device that supports two sets of planes. The other set of display planes is known as the overlay planes. These two sets of planes are treated as two separate screens in stacked mode and one screen in combined mode.

**input device**
Any of several pieces of equipment used to give information to the system. Examples are the keyboard, a mouse, or a digitizer tablet.

**keyboard binding**
Association of a special key press with a window manager function. For example, pressing the special keys (Shift) (Esc) displays the system menu of the active window.

**label**
The text part of an icon.

**local access**
The ability to run a program on the computer you are currently operating. This is different from remote access, where you run a program on a computer that is physically removed from the one you are operating.

**local client**
A local client is a program that is running on your `local` computer, the same system that is running your X server.

**mask**
A graphical image used in conjunction with another graphical element to hide unwanted graphical effects.

**matte**
A border located just inside the window between the client area and the frame. It is used to create a three-dimensional effect for the frame and window.

**menu**
A list of selections from which to make a choice. In a graphical user interface such as the X Window System, menus enable you to control the operation of the system.

**minimize**
To turn a window into an icon. The terms minimize and iconify are interchangeable.

**modifier key**
A key that, when pressed and held along with another key, changes the meaning of the other key. CTRL, Extend char, and Shift are examples of a modifier key.

**mouseless operation**
Although a mouse makes it easy to use the X Window System, the mouse is not absolutely necessary. The system can be configured to run from the keyboard alone.

**multi-tasking**
The ability to execute several programs (tasks) simultaneously on the same computer.

**node**
An address used by the system. For example, each device on the system has its own node. The system looks there whenever it needs to access the device. A node can also be an address on a network, the location of a system.

**non-client**
A program that is written to run on a terminal and so must be "fooled" by a terminal emulation window into running in the window environment.

**normalize**
To change an icon back into its "normal" (original) appearance. The opposite of iconify.

**overlay planes**
The secondary set of display planes on a device that supports two sets of planes. The other set of display planes is known as the image planes. These two sets of planes are treated as two separate screens.

**parent window**
A window that causes another window to appear. A window that "owns" other windows.

**pixel**
Short for "picture element." The individual dots, or components, of a screen. They are arranged in rows and columns and form the images that are displayed on the screen.

**pixmap**
An array of data bits used for graphics images. Each pixel (picture element) in the map can be several bits deep, resulting in multi-color graphics images.

**pointer**
Sometimes called the "mouse cursor," the pointer shows the location of the mouse. The pointer's shape depends on its location. In the root window, the pointer is an ×. On a window frame, the pointer is an arrowhead. Inside the frame, the pointer can be an arrowhead (as when it is inside a clock or load histogram frame) or an I-beam (as when it is inside a terminal window).

**press**
Strictly speaking, to hold down a mouse button or a key. Note that to hold down a mouse button *and move* the mouse is called "dragging."

**print server**
A computer that controls spooling and other printer operations. This permits a large number of individuals to efficiently share printer resources.

**remote access**
The ability to run a program on a computer that is physically removed from the one you are currently operating. This is different from local access, where you run a program on the computer that you are operating.

**remote client**
An X program that is running on a remote system, but the output of the program can be viewed on your terminal.

**remote host**
A computer physically removed from your own that you can log in to. See chapter 4 for prerequisites for establishing a remote host.

**resource**
That which controls an element of appearance or behavior. Resources are usually named for the elements they control.

**restoring**
The act of changing an minimized (iconified) or maximized window back to its regular size. The terms restoring and normalizing are usually interchangeable.

**root menu**
The menu associated with the root window. The root menu enables you to control the behavior of your environment.

**root window**
The root window is what the "screen" (the flat viewing surface of the terminal) becomes when you start X. To a certain extent, you can think of the root as the screen. The root window is the backdrop of your X environment. Although you can hide the root window under terminal windows or other graphic objects, you can never position anything behind the root window. All windows and graphic objects appear "stacked" on the root window.

**scalable fonts**
Scalable fonts are defined by a file containing a mathematical outline used by the system to create a bitmapped font for a particular size, slant, or weight.

**screen**
The physical CRT (Cathode Ray Tube) that displays information from the computer.

**screen dump**
An operation that captures an image from your screen, saves it in a file, and enables you to send that file to a printer for hardcopy reproduction.

**server**
A program that controls all access to input devices (typically a mouse and a keyboard) and all access to output devices (typically a display screen). It is an interface between application programs you run on your system and the system input and output devices.

**stacked mode**
A combination of image and overlay planes in which a single display has two "logical" screens, one the image planes, the other the overlay planes. Typically, the image planes are used to display graphics while the overlay planes are used to display text.

**system menu**
The menu that displays when you press the system menu button on the HP Window Manager window frame. Every window has a system menu that enables you to control the size, shape, and position of the window.

**Term0**
An HP level 0 terminal. It is a reference standard that defines basic terminal functions. For more information, see *Term0 Reference* in the HP-UX documentation set.

**terminal-based program**
A program (non-client) written to be run on a terminal (not in a window). Terminal-based programs must be "fooled" by terminal-emulation clients to run on the X Window System.

**terminal emulator**
A client program that provides a window within which you can run non-client programs. The non-client program runs just as though it were running from a real terminal rather than a window acting as a terminal.

**terminal type**
The type of terminal attached to your computer. HP-UX uses the terminal type to set the `TERM` environment variable so that it can communicate with the terminal correctly. The terminal type is usually set at login, but can be set afterward.

**terminal window**
A terminal window is a window that emulates a complete display terminal. Terminal windows are typically used to "fool" non-client programs into believing they are running in their favorite terminal—not a difficult task in most cases. When not running programs or executing operating system commands, terminal windows display the command-line prompt. Two terminal windows are supplied with X11—`hpterm`, which emulates HP terminals, and `xterm`, which emulates DEC and Tektronix terminals.

**text cursor**
The line-oriented cursor that appears in a terminal window after the command prompt. The term is used to distinguish the cursor used by a window from the cursor used by the mouse, the pointer.

**tile**
A rectangular area used to cover a surface with a pattern or visual texture. The HP Window Manager supports tiling, enabling users with limited color availability to create new color tiles blended from existing colors.

**title bar**
The title bar is the rectangular area between the top of the window and the window frame. The title bar contains the title of the window object, usually "Terminal Emulator" for `hpterm` windows, "xclock" for clocks, and "xload" for load histograms.

**transient window**
A window of short duration such as a dialog box. The window is only displayed for a short time, usually just long enough to get some direction from the user.

**window**
A data structure that represents all or part of the CRT display screen. It contains a two-dimensional array of 16-bit character data words, a cursor, a set of current attributes, and several flags. Visually, a window is represented as a rectangular subset of the display screen.

**window-based program**
A client or program written for use with the X Window System. The "opposite" of a window-based program is a terminal-based program.

**window decoration**
The frame and window control buttons that surround windows managed by the HP Window Manager.

**window manager**
The window manager controls the size, placement, and operation of windows on the root window. The window manager includes the functional window frames that surround each window object as well as a menu for the root window.

# Index

HEWLETT PACKARD

Reorder No.
B1171-90064

**Manufacturing
Part No.
B1171-90064**