Open Desktop Administrator's Guide

OPEN DESKTOP

Open Desktop™ Administrator's Guide

The Complete Graphical Operating System

OPEN DESKTOP

Open Desktop Administrator's Guide

OPEN DESKTOP™ Software

The copyrighted software that accompanies this manual is licensed to the End User only for use in strict accordance with the End User License Agreement, which License should be read carefully before commencing use of the software.

USE, DUPLICATION, OR DISCLOSURE BY THE UNITED STATES GOVERNMENT IS SUBJECT TO RESTRICTIONS AS SET FORTH IN SUBPARAGRAPH (c)(1) OF THE COMMERCIAL COMPUTER SOFTWARE -- RESTRICTED RIGHTS CLAUSE AT FAR 52.227-19 OR SUBPARAGRAPH (c)(1)(ii) OF THE RIGHTS IN TECHNICAL DATA AND COMPUTER SOFTWARE CLAUSE AT DFARS 52.227-7013. "CONTRACTOR/MANUFACTURER" IS THE SANTA CRUZ OPERATION, INC., 400 ENCINAL STREET, P.O. BOX 1900, SANTA CRUZ, CALIFORNIA 95061, U.S.A.

OPEN DESKTOP contains software licensed from a number of sources. The following are copyright notices for the software from these contributors which is used in OPEN DESKTOP.

OPEN DESKTOP Administration and User Documentation

Processed: Wed Dec 20 18:09:01 PST 1989
Document number : 12/21/89 1.0.0B

Administrator's Guide Contents

Administering ODT-VIEW

Administering ODT-OS

Administering ODT-NET

Administering ODT-DOS

Administering ODT-DATA

Administering ODT-VIEW

ODT-VIEW is based on technology developed for the X Window System by MIT, and technology developed for Motif by the Open Software Foundation, and technology developed for Xsight and Xhibit by Locus Computing Corporation.

12/21/89-1.0.0D
Processed: Wed Dec 20 11:38:40 PST 1989

Contents

Chapter 1
Introduction

When you first log in to Open Desktop™, the windows that you see on your screen are created by the X Window System, which controls and coordinates a series of window-generating programs. Whenever you open a new window, you start another program under the X Window System.

One such program, called the Desktop Manager, controls the appearance and behavior of the Desktop and the directory windows. Some of the components controlled by the Desktop Manager are: icon pictures and titles, mouse pointer appearance, and directory window frame buttons. Because the Desktop and the directory windows behave differently from other windows, you can configure the Desktop Manager separately from any other program running under the X Window System. Another window-generating program that runs under the X Window System is the Motif Window Manager. This program lets you control the aspects of your windows' appearance and behavior that are not controlled by the Desktop Manager.

You can change the characteristics of any Open Desktop window by changing the settings in the WINDOW CONFIGURATION FILES. The first part of this guide explains how to edit the configuration files for windows other than the Desktop and directory windows. This part contains the following chapters:

- Chapter 1, "Introduction"
- Chapter 2, "X Window System Overview"
- Chapter 3, "The .Xdefaults File"
- Chapter 4, "The .mwmrc File"

These chapters explain how to specify overall window characteristics such as:

- Color, size, and shape
- Focus policies
- Key and button bindings

Chapter 2,"X Window System Overview," also explains the steps that you must perform before you can edit the window configuration files.

The second part of this guide explains how to configure the Desktop Manager to modify the appearance and behavior of the Desktop and the directory windows. The following chapters are in Part II:

- Chapter 5, "Desktop Manager Overview"
- Chapter 6, "Desktop Manager Tutorials"
- Chapter 7, "Desktop Manager Reference"

These chapters explain how to control the following characteristics:

- How icons appear on the Desktop window
- What happens when you drop an icon
- How icons are chosen and activated
- File selection and manipulation

The appendixes at the end of this guide are organized as follows:

- Appendix A, "Setting Streams Parameters," explains how to allocate operating system resources for use with the X Window System.
- Appendix B, "Monochrome Configuration File," provides a sample *.Xdefaults* file for configuring a monochrome monitor.
- Appendix C, "Customizing Screen Colors," provides a sample *.Xdefaults* file for configuring a color monitor and describes how to modify screen display colors.
- Appendix D, "Changing Video Systems," explains how to reconfigure Open Desktop whenever you change monitors or video adaptor cards.

Chapter 2
X Window System Overview

This chapter explains how the X Window System is organized and describes the steps you need to take before actually editing the window configuration files. It also contains an overview of window frame construction and window focus policy. This chapter does not contain information about the Desktop Manager. An explanation of the Desktop Manager begins with Chapter 5, "Desktop Manager Overview."

X Window System Organization

The X Window System lets you communicate with Open Desktop through one or more windows displayed on your screen. The characteristics of each window are controlled by a set of default window configuration files. By changing the settings in these files, you can control:

- Window size, color, and shape
- Icon size, color, and shape
- Button and key bindings
- Menu characteristics
- Mouse behavior
- Focus policies

When you configure the X Window System, you are not limited to specifying just one set of characteristics for all windows. If you want to, you can create a unique look and feel for each application by giving each one a different set of window configuration settings.

Servers and Clients

The two fundamental parts of the X Window System are SERVERS and CLIENTS.

An X Window System server is a program that contains information about a particular workstation's hardware, such as the display, the keyboard, and the mouse. The server provides service; that is, it allows X Window System clients (applications) to open and close windows on your display. It also processes your input from the keyboard and the mouse.

An X Window System client is an application program, such as an editor, a database, a clock program, or the Desktop Manager.

Streams

The X Window System server and clients "talk" to each other using the UNIX® System V STREAMS MECHANISM. Some of your UNIX operating system's resources must be allocated to the streams mechanism. The speed with which Open Desktop runs depends on how economically its resources are used, so you should configure the streams parameters as judiciously as possible. Appendix A, "Setting Streams Parameters," explains how to estimate how much memory to reserve for streams, and how to perform the necessary adjustments.

The Window Manager

Because the server handles the specifics of a client's display, each client is hardware independent. In other words, a client's appearance and behavior are the same no matter what type of hardware you use. This adaptability is possible because client appearance and behavior are controlled by a special type of client called the WINDOW MANAGER. The window manager:

- Manages the resources that make up your windows
- Creates the frame around every window

The window manager included with the X Window System is the MOTIF WINDOW MANAGER, or mwm. When you configure the X Window System, you are actually changing the contents of the mwm configuration files.

Configuration Files

Mwm is configured from a database of resource specifications that control window appearance and behavior. The default resources are listed in */usr/lib/X11/app-defaults/Mwm*, which runs automatically whenever you log in to Open Desktop. To customize your windows, you must first copy */usr/lib/X11/app-defaults/Mwm* to *$HOME/.Xdefaults*, and then change the resource specifications as described in Chapter 3, "The .Xdefaults File."

Most specifications are directly controlled by either the */usr/lib/X11/app-defaults/Mwm* or *$HOME/.Xdefaults* file. However, there are several window attributes that require descriptions that are too detailed to be easily encoded in these files. A supplementary mwm RESOURCE DESCRIPTION FILE called */usr/lib/X11/system.mwmrc* describes these attributes, which control BUTTON BINDINGS, KEY BINDINGS, and MENU PANE DESCRIPTIONS. To customize this file, you must first copy it to *$HOME/.mwmrc*, and then change the resource specifications as described in Chapter 4, "The .mwmrc File."

The *.mwmrc* file is referenced by *.Xdefaults* whenever you log in to Open Desktop. It provides a convenient way to store several alternate specifications for button bindings, key bindings, or menu panes, which are then referenced by *.Xdefaults* when you log in. For example, you can define several styles of window panes in *.mwmrc*, and then specify in *.Xdefaults* which one is used when mwm starts up.

The configuration files that you create in your home directory have precedence over the default configuration files. If you delete the configuration files in your home directory, window control reverts back to */usr/lib/X11/app-defaults/Mwm* and *usr/lib/X11/system.mwmrc*.

Chapter 3, "The .Xdefaults File," describes each resource that can be set in *.Xdefaults*, provides syntax explanations for each resource group, and shows a sample *.Xdefaults* file.

Chapter 4, "The .mwmrc File," describes the attributes that you can set in *.mwmrc*, provides syntax explanations, and shows a sample *.mwmrc* file.

Window Frame Components

Default mwm window frames have the following components:

Table 2.1.

Window Frame Components

Component	Description
Title Area	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 the left mouse button and drag the window to a new location. 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.
Title Bar	The title bar includes the title area, the minimize button, the maximize button and the window menu button.
Resize Border Handles	To change the size of a window, move the pointer over a resize border handle (the cursor will change), 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.
Minimize Button	To turn the window back into its icon, do a left-button click on the minimize button (the frame box with a small square in it).
Maximize Button	To make the window fill the screen (or enlarge to the largest size allowed by the configuration files), do a left-button click on the maximize button (the frame box with a large square in it).
Window Menu Button	The window menu button is the horizontal bar in the frame box. To pop up the window menu, press the left mouse button while the pointer is on the horizontal bar. While pressing, drag the pointer to your selection on the menu and then release the button when your selection is highlighted. Alternately, you can click the left button on the bar to pop up the menu and then position the pointer and make your selection.
Matte	An optional matte decoration can be added between the client area and the window frame. There is no functionality associated with a matte.

Accelerator Keys

ACCELERATOR KEYS let you perform window manipulations from the keyboard. Most of the actions described in the previous table can be performed with accelerator keys. The accelerator keys and their functions are listed on the System menu.

Input Focus

By default, mwm supports a keyboard input focus policy of explicit selection. This policy specifies that when a window is selected to get keyboard input, it continues to get keyboard input until one of the following occurs:

- The window is withdrawn from window management
- Another window is explicitly selected to get keyboard input
- The window is iconified

The client window with the keyboard input focus has a visually distinctive window and frame.

The following tables summarize the keyboard input focus selection behavior:

Table 2.2.

Setting Focus with Buttons

Button Action	Object	Function
Button 1 press	Window / window frame	Selects keyboard focus
Button 1 press	Icon	Selects keyboard focus

Table 2.3.

Setting Focus with Keys

Key Action	Function
[Alt][Tab]	Moves input focus to next window in window stack.
[Alt][Shift][Tab]	Moves input focus to previous window in window stack.

Selecting Startup Clients

When you log in to Open Desktop, the **startx** command is automatically invoked. This command calls up */usr/lib/X11/sys.startxrc*, which contains the default list of X clients that are run every time you log in. To customize this list, you must first copy */usr/lib/X11/sys.startx* to *$HOME/.startxrc*, and then change the list to include the desired clients. Each line in *.startxrc* can contain only one client name, and you must place an ampersand (&) after all but the last client name in the file. Placing an ampersand after a client name specifies that that client is run in the background. Because the last client is not followed by an ampersand, it is run in the foreground.

$HOME/.startxrc must always contain "mwm," which is the window manager client.

NOTE: If a client is in a directory other than */usr/bin/X11*, you must give its full pathname when you list it in *.startxrc*.

Chapter 3
The .Xdefaults File

This chapter explains how the *.Xdefaults* file is organized, which resources it controls, and how you can reconfigure it.

.Xdefaults Overview

The following two sections explain how *.Xdefaults* resources are grouped, and how you can control either a single resource or an entire class of resources with a single specification.

Resource Organization

The mwm resources that are set in *.Xdefaults* are divided into the following categories:

Table 3.1.

Mwm Resource Categories

Resource	Description
Specific appearance and behavior	Lets you specify overall mwm appearance and behavior, such as keyboard and mouse behavior, icon size and placement, focus policies, and window frame size and shape. These resources do not control individual mwm components such as color or font style.
Component appearance	Lets you control the appearance of window manager menus, client window frames, and icons. Pixmaps, colors, and fonts are the most commonly configured component appearance resources.
Client specific	Lets you control the appearance and behavior of the windows associated with a client or class of clients. You can use these resources to give a different look-and-feel to each client that you run under Open Desktop.

Instances and Classes

Every resource in the *.Xdefaults* file belongs to a RESOURCE CLASS. Classes are composed of one or more RESOURCE INSTANCES. For example, the **Foreground** class contains the following resource instances: **foreground, bottomShadowColor, activeBottomShadowColor, activeForeground, iconImageBottomShadowColor, iconImageForeground, matteBottomShadowColor,** and **matteForeground.**

Setting the **Foreground** class to *blue* automatically sets each instance to blue. To set an instance to a value other than the one specified for its class, simply define the instance with the desired value. Instance specifications have precedence over class specifications, so the class setting is overridden in the case of an individually set instance. For example, setting the **matteForeground** instance to *yellow* and the **Foreground** class to *blue* produces a yellow matte foreground, with all other **Foreground** instances displayed in blue.

mwm Resource Descriptions and Syntax

The following is a sample *.Xdefaults* file. It contains examples of specific, component, and client appearance resources. It is not identical to your own *.Xdefaults* file, but gives you a general idea of what a *.Xdefaults* file should look like. The sections following the file describe the syntax for each type of appearance resource.

Sample .Xdefaults File

```
# SAMPLE .Xdefaults / app-defaults  RESOURCE SPECIFICATIONS FOR MWM
#


#
# general appearance resources that apply to mwm (all parts)
#

Mwm*font:                               hp8.8x16b

Mwm*backgroundTile:                     background

Mwm*activeForeground:                   Black
Mwm*activeBackground:                   Cyan
Mwm*activeTopShadowColor:               LightCyan
Mwm*activeBottomShadowColor:            Black
Mwm*makeActiveColors:                   false

Mwm*foreground:                         Black
Mwm*background:                         Gray
Mwm*topShadowColor:                     LightCyan
Mwm*bottomShadowColor:                  Black
Mwm*makeColors:                         false
#
Mwm*Xm*foreground:                      Red
Mwm*Xm*background:                      Green
Mwm*Xm*topShadowColor:                  White
Mwm*Xm*bottomShadowColor:               DarkSlateGray
Mwm*Xm*makeColors:                      false
```

Comment lines

Resource names

Resource values

```
# general appearance resources that apply to specific parts of mwm
#

Mwm*menu*background:                    LightCyan
Mwm*menu*topShadowColor                 Black
Mwm*menu*makeColors:                    false

#
# mwm - specific appearance and behavior resources
#

#Mwm*keyboardFocusPolicy:               pointer

Mwm*moveThreshold:                      40

Mwm*useIconBox:                         true


#
# xterm general appearance resources
#
xterm*background:White
xterm*foreground:Black


#
# Xhibit general appearance resources
#

# xhibit.geometry :                     +0+0
xhibit.desktop.geometry :               +0+0
xhibit.desktop.icon.titleGravity :      Top
xhibit.desktop.backgroundPixmap :       White.px
# xhibit.directory.backgroundPixmap :   White.px
xhibit.desktop.directory.background :   Cyan


#
# General appearance and behavior defaults
#
```

```
*topShadowTile:                 foreground
*bottomShadowTile:              foreground
*topShadowColor:                LightCyan
*bottomShadowColor:             Cyan
*foreground:                    Black
*background:                    White
*selectColor:                   Gray
*invertOnSelect:                true
*borderWidth:                   1
*borderColor:                   LightCyan


#
#  END OF RESOURCE SPECIFICATIONS
#
```

Specific Appearance and Behavior Resources

The syntax for selecting specific appearance and behavior resources is:

Mwm*resource_id: value*

For example, **Mwm*keyboardFocusPolicy: pointer** specifies that the keyboard focus moves to the window that contains the pointer.

Refer to the sample *.Xdefaults* file for more usage examples.

The following specific appearance and behavior resources can be specified:

Table 3.2.

Specific 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	NULL
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.	500
enforceKeyFocus	EnforceKeyFocus	T/F	T
execShell	ExecShell	string	SHELL
fadeNormalIcon	FadeNormalIcon	T/F	F
frameBorderWidth	FrameBorderWidth	pixels	5
iconAutoPlace	IconAutoPlace	T/F	T
iconBoxGeometry	IconBoxGeometry	string	6x1+0-0
iconBoxName	IconBoxName	string	iconbox
iconBoxTitle	IconBoxTitle	string	Icons
iconClick	IconClick	T/F	T
iconDecoration	IconDecoration	string	varies
iconImageMaximum	IconImageMaximum	wxh	50x50
iconImageMinimum	IconImageMinimum	wxh	32x32
iconPlacement	IconPlacement	string	left bottom
iconPlacementMargin	IconPlacementMargin	pixels	varies
interactivePlacement	InteractivePlacement	T/F	F
keyBindings	KeyBindings	string	system

(Continued on next page.)

Table 3.2.

Specific Appearance and Behavior Resources (*Continued*)

Name	Class	Value Type	Default
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
passButtons	PassButtons	T/F	F
passSelectButton	PassSelectButton	T/F	T
positionIsFrame	PositionIsFrame	T/F	T
positionOnScreen	PositionOnScreen	T/F	T
quitTimeout	QuitTimeout	millisec.	1000
resizeBorderWidth	ResizeBorderWidth	pixels	10
resizeCursors	ResizeCursors	T/F	T
showFeedback	ShowFeedback	string	all
startupKeyFocus	StartupKeyFocus	T/F	T
transientDecoration	TransientDecoration	string	system title
transientFunctions	TransientFunctions	string	-minimize -maximize
useIconBox	UseIconBox	T/F	F
wMenuButtonClick	WMenuButtonClick	T/F	T
wMenuButtonClick2	WMenuButtonClick2	T/F	T

autoKeyFocus (class **AutoKeyFocus**)
This resource is only available 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. The default value is true.

autoRaiseDelay (class **AutoRaiseDelay**)
This resource is only available 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 waits before raising a window after it gets the keyboard focus. The default value of this resource is 500 ms.

bitmapDirectory (class **BitmapDirectory**)
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*.

buttonBindings (class **ButtonBindings**)
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 NULL (that is, no button bindings are added to the built-in button bindings).

cleanText (class **CleanText**)
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.

clientAutoPlace (class **ClientAutoPlace**)
This resource determines the position of a window when the window has not been given a user-specified position. With a value of true, which is the default value, 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 attempts to place the windows completely on screen. The default value is true.

colormapFocusPolicy (class **ColormapFocusPolicy**)
This resource indicates the colormap focus policy that is to be used. If the resource value is explicit, then a colormap selection action is done on a client window to set the colormap focus to that window. If the value is pointer, then the client window containing the pointer has the colormap focus. If the value is keyboard, then the client window that has the keyboard input focus has the colormap focus. The default value for this resource is keyboard.

configFile (class **ConfigFile**)
The resource value is the pathname for an mwm resource description file. The default is *.mwmrc* in the user's home directory (based on the $HOME environment variable) if this file exists. Otherwise, it is */usr/lib/X11/system.mwmrc*.

deiconifyKeyFocus (class **DeiconifyKeyFocus**)
This resource only applies when the KeyboardFocusPolicy 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.

doubleClickTime (class **DoubleClickTime**)
This resource is used to set the maximum time (in milliseconds) between the clicks (button presses) that make up a double-click. The default value of this resource is 500 ms.

enforceKeyFocus (class **EnforceKeyFocus**)
If this resource has 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 value is false, the keyboard input focus is not explicitly set to globally active windows. The default value is true.

execShell (class **ExecShell**)
This resource lets you specify which shell is used when mwm executes programs from menus. By default, mwm uses the shell listed in the SHELL environment variable.

fadeNormalIcon (class **FadeNormalIcon**)
If this resource has a value of true, an icon is displayed in gray whenever it has been normalized (its window has been opened). The default value is false.

frameBorderWidth (class **FrameBorderWidth**)
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 5 pixels.

iconAutoPlace (class **IconAutoPlace**)
This resource indicates whether icons are automatically placed on the screen by mwm, or are placed by the user. Users may specify an initial icon position and may move icons after initial placement; however, mwm adjusts the user-specified position to fit into an invisible grid. When icons are automatically placed, mwm places them into the grid using a scheme set with the **iconPlacement** resource. If **iconAutoPlace** has a value of true, mwm carries out automatic icon placement. A value of false allows user placement. The default value of this resource is true.

iconBoxGeometry (class **IconBoxGeometry**)
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:

[=][*width*x*height*][{+-}*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.

iconBoxName (class **IconBoxName**)
This resource specifies the name that is used to look up icon box resources. The default name is iconbox.

iconBoxTitle (class **IconBoxTitle**)
This resource specifies the name that is used in the title area of the icon box frame. The default value is Icons.

iconClick (class **IconClick**)
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.

iconDecoration (class **IconDecoration**)
This resource specifies the general icon decoration. The resource value is label (only the label is displayed), image (only the image is displayed), or label image (both the label and image 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 value for icon box icons is label image. The default value for stand-alone icons is activelabel label image.

iconImageMaximum (class **IconImageMaximum**)
This resource specifies the maximum size of the icon image. The resource value is *width*x*height* (for instance, 64x64). The maximum supported size is 128x128. The default value of this resource is 50x50.

iconImageMinimum (class **IconImageMinimum**)
This resource specifies the minimum size of the icon image. The resource value is *width*x*height* (for instance, 32x50). The minimum supported size is 16x16. The default value of this resource is 32x32.

iconPlacement (class **IconPlacement**)
This resource specifies the icon placement scheme to be used. The resource value has the following syntax:

primary_layout secondary_layout

The layout values are shown in the following table:

Table 3.3.

Icon Layout Values

Name	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 layout value should not be used for both the *primary_layout* and the *secondary_layout* (for example, do not 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).

iconPlacementMargin (class **IconPlacementMargin)**
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 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).

interactivePlacement (class **InteractivePlacement)**
This resource controls the initial placement of new windows on the screen. If the value is true, then 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, then windows are placed according to the initial window configuration attributes. The default value of this resource is false.

keyBindings (class **KeyBindings)**
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 the mwm resource description file. The default value for this resource is the set of system-compatible key bindings.

keyboardFocusPolicy (class **KeyboardFocusPolicy**)
If this resource is set to pointer, the keyboard focus is 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 keyboard focus is 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.

limitResize (class **LimitResize**)
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.

lowerOnIconify (class **LowerOnIconify**)
If this resource has the value of true, which is the default value, 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 in the same place as its associated window.

maximumMaximumSize (class **MaximumMaximumSize**)
This resource limits 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.

moveThreshold (class **MoveThreshold**)
This resource controls 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 provision prevents window/icon movement when a click or double-click is done and there is unintentional pointer movement with the button down. The default value of this resource is 4 pixels.

passButtons (class **PassButtons**)
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, then 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.

passSelectButton (class **PassSelectButton**)
This resource indicates whether or not the keyboard input focus selection button press (if **keyboardFocusPolicy** is explicit) is passed on to the client window or is used to do a window management action associated with the window decorations. If the resource value is false, the button press is not used for any operation other than selecting the window that is to have the keyboard input focus. If the value is true, the button press is passed to the client window or used to do a window management operation, if appropriate. The keyboard input focus selection is done in either case. The default value for this resource is true.

positionIsFrame (class **PositionIsFrame**)
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.

positionOnScreen (class **PositionOnScreen**)
This resource indicates whether 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 then the size of the screen, at least the upper left corner of the window is on-screen. If the resource value is false, the windows are placed in the requested position even if they are completely off-screen. The default value of this resource is true.

quitTimeout (class **QuitTimeout**)
This resource specifies the amount of time (in milliseconds) that mwm waits for a client to update the WM_COMMAND property after mwm has sent the WM_SAVE_YOURSELF message. This protocol is only used for those clients that have a WM_SAVE_YOURSELF atom and no WM_DELETE_WINDOW atom in the WM_PROTOCOLS client window property. The default value of this resource is 1000 ms. (Refer to the f.kill function in Chapter 4, "The .mwmrc File," of this guide for additional information.)

resizeBorderWidth (class **ResizeBorderWidth**)
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 is 10 pixels.

resizeCursors (class **ResizeCursors**)
This resource indicates whether the resize cursors are always displayed when the pointer is in the window size border. If the value is true, the resize cursors are shown. Otherwise, the window manager cursor is shown. The default value is true.

showFeedback (class **ShowFeedback**)
This resource controls when feedback information is displayed. It controls both window position and size feedback during move or resize operations and initial client placement. It also controls window manager message and dialog boxes. The value for this resource is a list of names of the feedback options to be enabled; the names must be separated by a space. The names of the feedback options are shown in the following table:

Table 3.4.

Feedback Options

Name	Description
all	Shows all feedback. (Default value.)
behavior	Confirms behavior switch.
move	Shows position during move.
none	Shows no feedback.
placement	Shows position and size during initial placement.
resize	Shows size during resize.
restart	Confirms 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.

startupKeyFocus (class **StartupKeyFocus**)
This resource is only available 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).

transientDecoration (class **TransientDecoration**)
This resource 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 resize borders and a title bar with a window menu button).

transientFunctions (class **TransientFunctions**)
This resource indicates 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.

useIconBox (class **UseIconBox**)
If this resource has 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).

wMenuButtonClick (class **WMenuButtonClick**)
This resource indicates whether a click of the mouse when the pointer is over the window menu button posts and leaves posted the system menu. If this resource has a value of true, the menu remains posted. True is the default value for this resource.

wMenuButtonClick2 (class **WMenuButtonClick2**)
When this resource has a value of true, a double-click action on the window menu button performs an f.kill function. The default value of this resource is true.

Component Appearance Resources

The syntax for specifying component appearance resources is:

Mwm*resource_id: value*

For example, **Mwm*foreground: VioletRed** specifies that VioletRed is the foreground color for mwm menus, icons, and client window frames.

The syntax for specifying component appearance resources that apply to a particular mwm component is:

Mwm*[**menu**|**icon**|**client**|**feedback**]**resource_id: value*

If *menu* is specified, the resource applies only to mwm menus; if *icon* is specified, the resource applies to icons; and if *client* is specified, the resource applies to client window frames. For example, **Mwm*icon*foreground** specifies the foreground color for mwm icons; **Mwm*menu*foreground** specifies the foreground color for mwm menus; and **Mwm*client*foreground** specifies 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: value*

For example, **Mwm*client*title*foreground: red** specifies that red is 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:** *value*

For example, **Mwm*menu*my_menu*foreground: red** specifies that red is the foreground color for the menu named **my_menu**.

Refer to the sample *.Xdefaults* file for more usage examples.

The following table lists the component appearance resources that apply to all window manager parts.

Table 3.5.

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.
**Pixmap image name.
***X11 R3 Font description.

background (class **Background**)
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.

backgroundPixmap (class **BackgroundPixmap**)
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 based on the visual type of the screen.

bottomShadowColor (class **Foreground**)
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.

bottomShadowPixmap (class **BottomShadowPixmap**)
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.

fontList (class **Font**)
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 value is fixed.

foreground (class **Foreground**)
This resource specifies the foreground color. The default is chosen based on the visual type of the screen.

saveUnder (class **SaveUnder**)
This resource indicates whether "save unders" are used for mwm components. For this resource 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 sets the save under attribute on the window manager frame of any client that has it set. If **saveUnder** is false, save unders are not used on any window manager frames. The default value is false.

topShadowColor (class **Background**)
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.

topShadowPixmap (class **TopShadowPixmap**)
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 based on the visual type of the screen.

The following table lists the component appearance resources that apply to frame and icon components:

Table 3.6.

Component Appearance Resources — 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).

activeBackground (class **Background**)
This resource specifies the background color of the mwm decoration when the window is active (has the keyboard focus). The default is based on the visual type of the screen.

activeBackgroundPixmap (class **ActiveBackgroundPixmap**)
This resource specifies the background Pixmap of the mwm decoration when the window is active (has the keyboard focus). The default is based on the visual type of the screen.

activeBottomShadowColor (class **Foreground**)
This resource specifies the bottom shadow color of the mwm decoration when the window is active (has the keyboard focus). The default is based on the visual type of the screen.

activeBottomShadowPixmap (class **BottomShadowPixmap**)
This resource specifies the bottom shadow Pixmap of the mwm decoration when the window is active (has the keyboard focus). The default is based on the visual type of the screen.

activeForeground (class **Foreground**)
This resource specifies the foreground color of the mwm decoration when the window is active (has the keyboard focus). The default is based on the visual type of the screen.

activeTopShadowColor (class **Background)**
This resource specifies the top shadow color of the mwm decoration when the window is active (has the keyboard focus). The default is based on the visual type of the screen.

activeTopShadowPixmap (class **TopShadowPixmap)**
This resource specifies the top shadow Pixmap of the mwm decoration when the window is active (has the keyboard focus). The default is based on the visual type of the screen.

Client-Specific Resources

The syntax for specifying client-specific resources is:

Mwm*client_name_or_class***resource_id:** *value*

For example, **Mwm*mterm*windowMenu: ClientsMenu** specifies that the menu called "ClientsMenu" is the window menu used with mterm clients.

The syntax for specifying client-specific resources for all classes of clients is:

Mwm*resource_id:* *value*

Specific-client specifications take precedence over the specifications for all clients. For example, **Mwm*windowMenu: DefaultSystemMenu** specifies that "DefaultSystemMenu" is the window menu for all classes of clients that do not have a specified window menu.

The syntax for specifying resource values for windows that have an unknown name and class is:

Mwm*defaults*resource_id:* *value*

For example, **Mwm*defaults*iconImage: /usr/lib/X11/generic.icon** specifies that the icon image in the file *generic.icon* is used for windows that have an unknown name and class.

Refer to the sample *.Xdefaults* file for more usage examples.

The following client-specific resources can be specified:

Table 3.7.

Client-Specific Resources

Name	Class	Value Type	Default
clientDecoration	ClientDecoration	string	all
clientFunctions	ClientFunctions	string	all
focusAutoRaise	FocusAutoRaise	T/F	T
iconImage	IconImage	pathname	(image)
iconImageBackground	Background	color	icon background
iconImageBottomShadowColor	Foreground	color	icon bottom shadow
iconImageBottomShadowPixmap	BottomShadowPixmap	color	icon bottom shadow pixmap
iconImageForeground	Foreground	color	icon foreground
iconImageTopShadowColor	Background	color	icon top shadow color
iconImageTopShadowPixmap	TopShadowPixmap	color	icon top shadow pixmap
matteBackground	Background	color	background
matteBottomShadowColor	Foreground	color	bottom shadow color
matteBottomShadowPixmap	BottomShadowPixmap	color	bottom shadow pixmap
matteForeground	Foreground	color	foreground
matteTopShadowColor	Background	color	top shadow color
matteTopShadowPixmap	TopShadowPixmap	color	top shadow pixmap
matteWidth	MatteWidth	pixels	0
maximumClientSize	MaximumClientSize	wxh	fill the screen
useClientIcon	UseClientIcon	T/F	F
windowMenu	WindowMenu	string	string

clientDecoration (class **ClientDecoration**)
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, then 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, then 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.

The following table describes the **clientDecoration** values:

Table 3.8.

Values for clientDecoration

Value	Description
all	Includes 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 line removes the resize handles and maximize button from XClock windows.

Mwm*XClock.clientDecoration: menu minimize border

This line does the same thing as the first example. Note that either **menu** or **minimize** implies **title.**

clientFunctions (class **ClientFunctions**)
This resource indicates 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, 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, 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 be separated from the next function by a space.

The following table lists the functions available for this resource:

Table 3.9.

Values for clientFunctions

Value	Description
all	Includes all functions (default value)
none	No functions
resize	f.resize
move	f.move
minimize	f.minimize
maximize	f.maximize
close	f.kill

focusAutoRaise (class **FocusAutoRaise**)
When the value of this resource is true, clients are made completely unobscured 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.

iconImage (class **IconImage**)
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 determines 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.

iconImageBackground (class **Background**)
This resource specifies the background color of the icon image that is displayed in the image portion of an icon. The default value of this resource is the icon background color (that is, specified by **Mwm*background** or **Mwm*icon*background**).

iconImageBottomShadowColor (class **Foreground**)
This resource specifies the bottom shadow color of the icon image that is displayed in the image portion of an icon. The default value of this resource is the icon bottom shadow color (that is, specified by **Mwm*icon*bottomShadowColor**).

iconImageBottomShadowPixmap (class **BottomShadowPixmap**)
This resource specifies the bottom shadow Pixmap of the icon image that is displayed in the image portion of an icon. The default value of this resource is the icon bottom shadow Pixmap (that is, specified by **Mwm*icon*bottomShadowPixmap**).

iconImageForeground (class **Foreground**)
This resource specifies the foreground color of the icon image that is displayed in the image portion of an icon. The default value of this resource is the icon foreground color (that is, specified by **Mwm*foreground** or **Mwm*icon*foreground**).

iconImageTopShadowColor (class **Background**)
This resource specifies the top shadow color of the icon image that is displayed in the image portion of an icon. The default value of this resource is the icon top shadow color (that is, specified by **Mwm*icon*topShadowColor**).

iconImageTopShadowPixmap (class **TopShadowPixmap**)
This resource specifies the top shadow Pixmap of the icon image that is displayed in the image portion of an icon. The default value of this resource is the icon top shadow Pixmap (that is, specified by **Mwm*icon*topShadowPixmap**).

matteBackground (class **Background**)
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**).

matteBottomShadowColor (class **Foreground**)
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**).

matteBottomShadowPixmap (class **BottomShadowPixmap**)
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**).

matteForeground (class **Foreground**)
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**).

matteTopShadowColor (class **Background**)
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"**).

matteTopShadowPixmap (class **TopShadowPixmap**)
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**).

matteWidth (class **MatteWidth**)
This resource specifies the width of the optional matte. The default value is 0, which effectively disables the matte.

maximumClientSize (class **MaximumClientSize**)
This is a size specification that indicates the client size to be used when an application is maximized. The resource value is specified as *width***x***height*. The width and height are interpreted in the units that the client uses (for example, with terminal emulators this is generally characters). 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.

useClientIcon (class **UseClientIcon**)
If the value for this resource is true, a client-supplied icon image takes precedence over a user-supplied icon image. The default value is false, making the user-supplied icon image have higher precedence than the client-supplied icon image.

windowMenu (class **WindowMenu**)
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**. The default value of this resource is the name of the built-in window menu specification.

Chapter 4
The .mwmrc File

This chapter explains how the *.mwmrc* file is organized, which window attributes it controls, and how you can reconfigure it.

.mwmrc Overview

The *.mwmrc* file is a supplementary resource description file that is referred to by *.Xdefaults*. It contains descriptions of resources that cannot be easily encoded in *.Xdefaults*. Usually, you only need one configurable supplementary resource description file (*$HOME/.mwmrc*) in addition to the default supplementary file (*/usr/lib/X11/system.mwmrc*). If you create more than one configurable supplementary file, you must use the **configFile** resource in *.Xdefaults* to specify which one is referenced when you log in to Open Desktop.

The *.mwmrc* file uses WINDOW MANAGER FUNCTIONS to define the behavior of the resource types shown in the following table. When you configure a resource in *.mwmrc*, you do so by assigning one or more window manager functions to it. These functions are explained in detail later in this chapter.

The following types of resources can be described in *.mwmrc*:

Table 4.1.

Configuration of .mwmrc Resource Types

Resource Type	How Resource Type Is Configured
Buttons	Window manager functions can be bound to mouse button events.
Keys	Window manager functions can be bound to key-press events.
Menu Panes	The contents of menu panes and the key-press or button events that post them can be defined.

Sample .mwmrc File

The *.mwmrc* is a standard text file containing items of information separated by blanks, tabs, and new-line characters. The following guidelines apply to the *.mwmrc* file:

- Blank lines are ignored.
- Items or characters that have special meaning are interpreted literally when quoted. For example, if you quote the comment character, it is not interpreted as the comment character.
- Items longer than one character are quoted with double quotes (").
- A single character is quoted by preceding it with a backslash (\).
- All text from an unquoted # to the end of the line is regarded as a comment.
- If ! is the first character in a line, the line is regarded as a comment.

The following sample *.mwmrc* file contains examples of window manager functions that control menu panes, key bindings, and button bindings. The sections following the file describe the functions and the syntax for using them to control these resource types.

```
#
#
#       @(#) system.mwmrc 1.2 89/04/04
#
#   DEFAULT mwm RESOURCE DESCRIPTION FILE (system.mwmrc)
#



#
# menu pane descriptions
#

Menu DefaultWindowMenu MwmWindowMenu
{
    Restore         _R      Alt<key>5       f.normalize
    Move            _M      Alt<key>7       f.move
    Size            _S      Alt<key>8       f.resize
    Minimize        _n      Alt<key>9       f.minimize
    Maximize        _x      Alt<key>0       f.maximize
```

Comment Lines

Menu name

Accelerator Keys

Functions

Menu labels

Mnemonics

```
    Lower               _w      Alt<key>minus          f.lower
    no-label                                           f.separator
    Close               _C      Alt<key>4              f.kill
}

Menu RootMenu
{
    "Root Menu"                         f.title
    "Clients"                   _C      f.menu ClientsMenu
    "Xterm"                     _x      f.exec "xterm -sb  &"
    "Shuffle Up"                _U      f.circle_up
    "Shuffle Down"              _D      f.circle_down
    "Refresh"                   _R      f.refresh
    no-label                            f.separator
    "Restart"                           f.restart
}

Menu ClientsMenu
{
    "xclock"                    _k      f.exec "xclock &"
    "xload"                     _d      f.exec "xload &"
    "xcalc"                     _l      f.exec "xcalc &"
    "xbiff"                     _f      f.exec "xbiff &"
    "bitmap"                    _p      f.exec "bitmap $HOME/tmp_bitmap &"
    "ico"                       _o      f.exec "ico &"
}


#
# key binding descriptions
#

Keys _DefaultKeyBindings_

{
    Shift<Key>Escape        icon|window         f.post_smenu
    Meta<Key>Escape         root|icon|window    f.menu DefaultRootMenu
    Meta Shift<Key>Tab      root|icon|window    f.prev_key
    Meta<Key>Tab            root|icon|window    f.next_key
}
```

Key binding set name

Key events

Contexts

Functions

```
#
# button binding descriptions
#

Buttons DefaultButtonBindings
{
    <Btn1Down>              root            f.menu   DefaultRootMenu
    <Btn3Down>              root            f.menu   DefaultRootMenu
    <Btn1Down>              frame           f.raise
    <Btn3Down>              frame|icon      f.post_smenu
    Meta<Btn1Down>          icon|window     f.move
    Meta<Btn3Down>          window          f.minimize
}

#
#  END OF mwm RESOURCE DESCRIPTION FILE
#
```

Button binding set name

Button events

Contexts

Functions

Window Manager Functions

As shown in the sample file, window manager functions are key components in describing menu panes, key bindings, and button bindings. The following three sections describe the functions, their syntax, and their constraints.

Function Descriptions

Each type of resource (menu panes, keys, or buttons) uses one or more window manager functions to control its behavior. For example, binding the left mouse button to a client window with the **f.raise** function causes the window to be raised whenever you press the left mouse button.

The following list describes each window manager function:

f.beep
This function causes a beep.

f.circle_down [icon | window]
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 is no longer obscuring any other window or icon).

This function affects only those windows and icons that are obscuring 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.

f.circle_up [**icon** | **window**]
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 are obscuring 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.

f.exec or **!**
This function causes a command to be executed (using 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.

f.focus_color
This function sets the colormap focus to a client window. If this function is done in a root context, then 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.

f.focus_key
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.

f.kill
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 and the WM_DELETE_WINDOW protocol is not 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.

f.lower [*-client*]
This function lowers a client window to the bottom of the window stack (where it obscures

no other window). Secondary windows (that is, transient windows) are restacked with their associated primary window. 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.

f.maximize
This function causes a client window to be displayed with its maximum size.

f.menu
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.

f.minimize
This function causes a client window to be minimized (iconified). A window is minimized when no icon box is used, and its icon is placed on the bottom of the window stack (such that it obscures no other window). If an icon box is used, then 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.

f.move
This function allows a client window to be interactively moved.

f.next_cmap
This function installs the next colormap in the list of colormaps for the window with the colormap focus.

f.next_key [icon | window | transient]
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 only moved to windows that do not have an associated secondary window that is application modal. If the *transient* argument is specified, then 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, then the function applies only to icons. If a window function argument is specified, then the function applies only to windows.

f.nop
This function does not cause any actions to be performed. When you want to include a command line that temporarily causes no action, you can use **f.nop** to satisfy the syntax requirement that a function of some type be named.

f.normalize
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.

f.pack_icons
This function redraws icons on the root window or in the icon box based on the layout policy being used. In general, this causes icons to be "packed" into the icon grid.

f.pass_keys
This function is used to enables/disables (toggles) the 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.

f.post_wmenu
This function posts the window menu. If a key posts 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.

f.prev_cmap
This function installs the previous colormap in the list of colormaps for the window with the colormap focus.

f.prev_key [icon | window | transient]
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 only moved to windows that do not have an associated secondary window that is application modal. If the *transient* argument is specified, then 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.

f.quit_mwm
This function terminates mwm (but not the X Window System).

f.raise [*-client*]
This function raises a client window to the top of the window stack (where it is obscured by no other window). Secondary windows (that is, transient windows) are restacked with their associated primary window. The *client* argument indicates the name or class of a client to raise. If the *client* argument is not specified, the context that the function was invoked in indicates the window or icon to raise.

f.raise_lower
This function raises a client window to the top of the window stack if it is partially obscured by another window; otherwise, it lowers the window to the bottom of the window stack. Secondary windows (that is, transient windows) are restacked with their associated primary window.

f.refresh
This function causes all windows to be redrawn.

f.refresh_win
This function causes a client window to be redrawn.

f.resize
This function allows a client window to be interactively resized.

f.restart
This function causes **mwm** to be restarted (effectively terminated and re-executed).

f.send_msg *message_number*
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 only sent 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.

f.separator
This function causes a menu separator to be put in the menu pane at the specified location (the label is ignored).

f.set_behavior
This function causes the window manager to restart with the default OSF behavior (if a custom behavior is configured) or a custom behavior (if an OSF default behavior is configured).

f.title
This function inserts a title in the menu pane at the specified location.

Function Syntax

There are two types of syntax described in this chapter: syntax for defining a resource type, and syntax for naming a function. Functions are a common component of every *.mwmrc* resource type description. Thus, while the syntax for describing resource types varies between resources, the syntax for naming a function is the same no matter what resource type the function describes. This section describes the syntax for naming a function. The different syntaxes for each resource type are described later in this chapter. The syntax for naming a function is:

function = *function_name* [*function_args*]
function_name = *window manager function*
function_args = {*quoted_item* | *unquoted_item*}

Refer to the sample *.mwmrc* file for examples of function usage.

Function Constraints

Some functions cannot be specified by certain resource types. For example, you cannot use the **f.title** function to define a button or key binding; you can only use it to define a menu pane. There are also constraints regarding the context in which a function can be used. For example, the **f.minimize** function only applies to window panes; it does not work when the pointer is on the root menu or an icon.

You can configure a function's context as long as you stay within the limits of that function's constraints. For example, you can configure the **f.kill** function to work with icons, windows, or both. However, because the root window is not an available context in which to use **f.kill**, you cannot configure it in that context. The following table describes the three contexts in which functions can be used:

Table 4.2.

Function Contexts

Context	Description
root	The function can be performed when: 1. The pointer is on the root menu, and 2. Neither a client window nor an icon is to be acted upon by the function.
icon	The function can be performed when the pointer is on an icon.
window	The function can be performed when the pointer is on a client window, title bar, or frame. Some functions, such as **f.maximize**, apply only when the window is normalized. Others, such as **f.normalize**, apply only when the window is maximized.

The following list describes how resource type names are used in table 4.4:

Table 4.3.

Resource Types

Resource Type	Definition
button	The function can be specified in the button bindings section of *.mwmrc*.
key	The function can be specified in the key bindings section of *.mwmrc*.
menu	The function can be specified in the menu pane description section of *.mwmrc*.

If a function is specified in an incompatible resource type, or if it is invoked in a context that does not apply, the function is treated as **f.nop**. The following table describes the contexts and resource types that work with each function:

Table 4.4.

Where Functions Can Be Used

Function	Contexts	Resource Types
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	root,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.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	button,key,menu

(Continued on next page.)

Table 4.4.

Where Functions Can Be Used (*Continued*)

Function	Contexts	Resource Types
f.raise	root,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.restart	root	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

Using Functions

The following sections describe how to use the window manager functions to configure menu panes, key bindings, and button bindings. The concept of WINDOW MANAGER EVENTS is also explained.

Window Manager Events

Events are another part of the specifications for menu pane descriptions, key binding sets, and button binding sets. An event describes an action that you take (such as pressing a mouse button) to execute a function. The next three sections explain how to link a function to an event. This section explains the event syntax that is used in the sections that follow.

The *button* event specification used later in this chapter in "Configuring Button Bindings" has the following syntax:

button = [*modifier_list*]<*button_event_name*>
modifier_list = *modifier_name* {*modifier_name*}

All modifiers specified are exclusive; that is, 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 for an event specification.

Table 4.5.

Modifiers

modifier_name	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*.

Table 4.6.

Button Event Definitions

button_event_name	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 are single key presses; key releases are ignored. The *key* event specification used later in this chapter in "Configuring Key Bindings" has the following syntax:

key = [*modifier_list*]<**Key**>*key_name*
modifier_list = *modifier_name* {*modifier_name*}

All modifiers are exclusive; that is, only the specified modifiers can be present when the key event occurs. Modifiers for keys are the same as those that apply to buttons.

Refer to the sample *.mwmrc* file for examples of window manager event usage.

Configuring Menu Panes

Menus can be popped up with the *f.post_wmenu* and *f.menu* window manager functions. The context for functions that are available through the newly displayed menu depends on how the menu was popped up. If the menu was popped up with a key-press event or from another menu, the function context is determined by the keyboard input focus. If the menu was popped up with a button-press event, the context of the button binding dictates 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. The *label* may be a string or a bitmap file. The label specification has the following syntax:

```
label =        text | bitmap_file
text =         quoted_item | unquoted_item
bitmap_file =  @file_name
```

The string encoding for labels must match the font that is used in the menu. Labels are grayed out for menu items that perform the **f.nop** function, an invalid function, or a function that is not available in the current context.

Mnemonics are functional only when the menu is posted. 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. The *character* must exactly match a character in the label; the mnemonic cannot execute if any modifier (such as Shift) is pressed with the character key.

The *accelerator* is a key event specification with the same syntax as the window manager function key bindings.

Refer to the "menu pane descriptions" section of the sample *.mwmrc* file for examples of menu pane descriptions.

Configuring Key Bindings

The **keyBindings** resource in *.Xdefaults* refers to a set of key bindings in *.mwmrc*. These bindings cause window manager functions to be performed when particular keys are pressed. The context in which each key binding applies is indicated in the key binding specification.

The key binding syntax is:

Keys *bindings_set_name*
{
  key context function
  key context function
  .
  .
  key context function
}

The syntaxes for *key* and *function* were explained earlier in this chapter in "Window Manager Events" and "Function Syntax," respectively.

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 key 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 key binding to be effective. The **title** context is for the title area of the window management frame, 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), and the **app** context is for the application window (not including the window management frame).

The context for a key event is the same as the context for the window or icon that has the keyboard input focus. If no window or icon has the keyboard input focus, the context is set to **root**. The **frame**, **title**, **border**, and **app** contexts are the same as the **window** context.

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 automatically uses the same key to remove the menu from the screen after it has been popped up.

Refer to the "key bindings descriptions" section of the sample *.mwmrc* file for examples of key bindings.

Configuring Button Bindings

The **buttonBindings** resource in *.Xdefaults* refers to a set of button bindings in *.mwmrc*. These button bindings cause window manager functions to be performed when a button press occurs with the pointer over a framed client window, an icon, or the root window. The contexts specified in the button binding definitions determine the contexts of the window manager functions that are made available when you perform the button press. For example, suppose that pressing the left mouse button pops up a menu. The context of each menu item's function is the same as the context of the function that bound the left mouse button to the menu pop-up.

The button binding syntax is:

```
Buttons bindings_set_name
{
  button   context   function
  button   context   function
              .
              .
  button   context   function
}
```

The syntaxes for *button* and *function* were explained earlier in this chapter in "Window Manager Events" and "Function Syntax," respectively.

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 **title** context is for the title area of the window management frame, 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), 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.

Refer to the "button binding description" section of the sample *.mwmrc* file for examples of button bindings.

Chapter 5
Desktop Manager Overview

The following three chapters describe how to:

- Reconfigure the Desktop Manager to modify the appearance and behavior of the Desktop window and its components
- Customize the Desktop Manager to suit particular applications

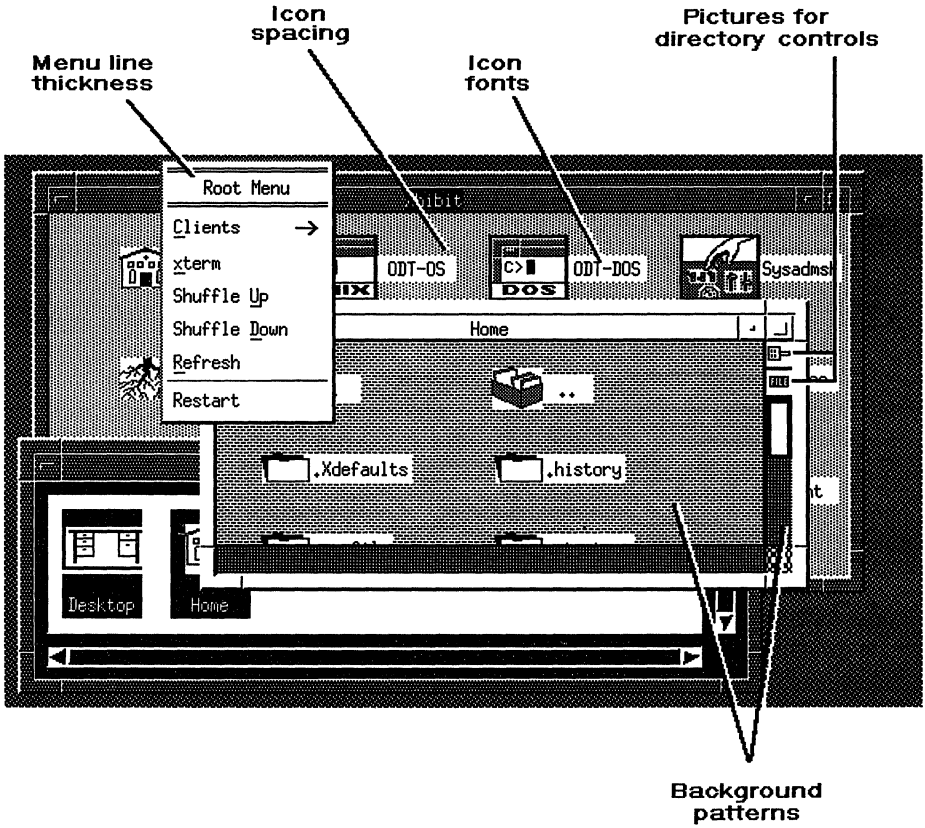
The most powerful feature of the Desktop Manager is that its appearance and behavior are not fixed, but are determined instead by rule files that can be edited to suit individual requirements. This chapter gives a general introduction to the DEFAULTS FILES and RULE FILES, which determine the behavior and appearance of the Desktop Manager. It also describes some typical applications in which these files are used.

Because the Desktop Manager is an X client, many of its rule and defaults files have names that begin with an "x," such as *.xdtuserinfo*.

Changing the Appearance of the Desktop Manager

The defaults files determine the default appearance of the main components of the Desktop Manager, as shown in the following figure:

Figure 5-1. Desktop Manager Characteristics Specified by Defaults Files



In addition, the defaults files specify the mapping between the mouse clicks and the triggers used by the system. These can be altered to accommodate a mouse with a different number or layout of buttons.

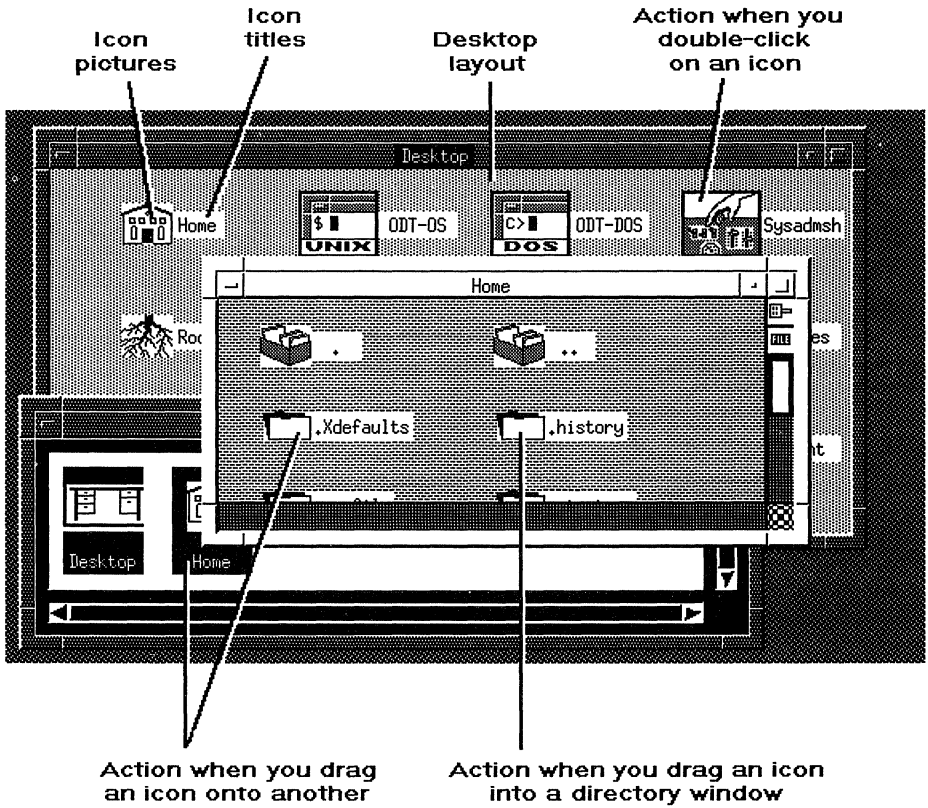
The system defaults file specifies the name of the startup environment file. This environment file contains the desktop layout, which is updated each time you leave the Desktop Manager. Normally, each computer running the Desktop Manager has a defaults file suited to the particular machine on which it is being run. For example, machines with large screens use larger pictures for window controls to improve legibility.

However, users can provide their own defaults files to give any desired appearance, or they can switch between defaults files to provide different working environments for different applications.

Changing the Behavior of the Desktop Manager

A separate set of files called rule files determine the characteristics of the Desktop Manager shown in the following figure:

Figure 5-2. Desktop Manager Characteristics Specified by Rule Files



The default behavior of the Desktop Manager is determined by a system rule file, but this can be overridden by additional rule files provided by each user. Furthermore, you can create rule files that are local to a specific directory. Rule files consist of a number of separate components that determine the behavior of the Desktop Manager:

- **Icon appearance and title.** The picture displayed for a file or group of files, and the title displayed beside it.
- **Icon activation.** This occurs when the icon is activated, or double-clicked, by the mouse, and when another icon is dropped onto it.
- **Drop behavior.** This happens when one or more icons are dragged into a directory window.
- **Desktop layout.** The files that are on the desktop and what their positions are.
- **Locked files.** The files that are permanently locked onto the desktop.

Typical Applications

The following examples illustrate how the Desktop Manager can be configured in some applications.

Simulating Familiar Environments

Users who are already familiar with other desktop-based systems, or who have to share their work between the Desktop Manager and another desktop system, can configure the Desktop Manager to match the appearance and behavior of the other system. Thus, they can minimize the errors associated with transfer and reduce the amount of relearning needed.

Associating Data Files with Programs

By defining appropriate icon rules, each data file can be associated with the most relevant program, so that double-clicking on the data file invokes the appropriate program with the data file supplied to it.

Dragging a data file onto a program icon can be used as an alternate method of invoking the program. For example, a rule file could be written such that compiler source files are edited by dragging them onto the editor icon and compiled by double-clicking on them.

Creating a Background Mail Server

The rule files are flexible enough to allow the creation of applications, such as a mail server that posts new mail as an icon on the desktop. The mail server would run as a background task, and when mail is received it would run a Desktop command language command to place the mail file on the desktop. The rule file could specify that double-clicking on a mail file opens it in a text editor and deletes the original file.

Waste Icon

The Waste icon is created by rule files with no additional programming. The Waste icon is a directory, with an appropriate title and picture. It contains a rule file that causes any icon either dropped into the directory window or onto the directory icon to be moved to the directory. The Waste directory can be cleared by double-clicking its icon with the right-most mouse button. The Waste icon is typically locked onto the desktop by including it in the locked files list.

Encrypting and Decrypting Automatically

The drop rule files allow directories with special characteristics to be created within the Desktop Manager. For example, a directory could be created such that all files dragged into it are encrypted. They could be decrypted either by dragging them into another directory window or by double-clicking on their icon.

Using Local Rule Files

Rule files can be local to one user, or even local to a specific directory. User-specific rule files can take care of the different computers that different users on a UNIX system network might be using. Each user can thus double-click on a program file on the network, and run it on a computer appropriate to that program.

Changing Environments

The Desktop Manager is flexible enough to allow a single user to switch environments at will, thus changing both the appearance and behavior of the desktop by double-clicking the appropriate environment file. For example, a user might have two characteristic modes of working, programming and documenting. In the first mode, the desktop might display compilers and debuggers, and the rule files could specify that double-clicking source files runs the appropriate compiler. In the documenting mode, the desktop might display word processing and flow-charting programs, and double-clicking source files loads the appropriate word processor.

The standard rules take any file ending in *.xde* as an environment file. Double-clicking an environment file with the left-hand mouse button changes the desktop to that environment.

Tailoring Message Files

The message file contains all the messages used by the Desktop Manager in a standard editable form. You can alter this file to tailor the messages to specific applications: for example, more explanatory messages in a teaching environment or terse messages in a development environment. Or, you can change the file to cater to different languages.

Chapter 6
Desktop Manager Tutorials

This chapter contains four examples that show how to configure the Desktop Manager to make a file compression utility available from the Desktop. These examples illustrate most of the main features of the Desktop Manager rule files and should serve as a useful basis for creating other applications.

Determining the Appearance of Your Desktop

By creating rule files, you can assign icons to specific files or directories, and you can specify a title for the icon to be displayed instead of the usual file title. The following example illustrates the use of rule files by defining a title and icon for the UNIX file compression utility, **compress**. Make a copy of the **compress** program in a suitable directory in your user file space.

Creating an Icon

First we need to define a suitable icon and put it into the picture file directory */usr/include/X11/bitmaps/desktop_icons*. The simplest procedure is to start with an existing picture file, as follows:

1. Copy an existing picture file and rename it *compress.px*.
2. Double-click its icon to run the bitmap editor. You can then edit the icon with the mouse pointer.
3. Exit from the bitmap editor by clicking Save and then Quit when you are satisfied with the picture.

In the bitmap editor, the mouse buttons have the following functions:

Table 6.1.

Bitmap Editor Button Bindings

Button	Function
Left	Sets pixel black.
Middle	Changes pixel.
Right	Sets pixel white.

Defining an Icon Rule

The next step is to define an icon rule assigning the appropriate picture and title to the **compress** utility. We give the utility the title Squash. Icon rules have the format:

ic { [*file-spec* { *file-rules* }] I }

where **ic** is a label identifying icon rules, and *file-spec* is a construct specifying the icons to which the rules should apply. In this example only the file compress is to be affected. *file-rules* gives a list of rules specifying the appearance and behavior of all the selected icons. In this example the rules are:

ti =Squash; pi = desktop_icons/compress.px

The label **ti** introduces a title for the icon; in this case **Squash**. The label **pi** introduces the picture to be displayed for the icon instead of its default picture; in this case, the one we defined in the bitmap editor. The full icon rule is thus:

ic { compress { ti =Squash; pi = desktop_icons/compress.px } }

Adding the Rule to a Rule File

Put this rule in a file named *.xdtdirinfo* in the directory containing **compress**, and the icon changes to display the picture and title you have designed. If this rule file already exists, you must incorporate the new icon rule into the rules it already contains. If there are no other icon rules in the file starting with **ic**, you can put this rule at the end of the file as it stands. Otherwise, insert this rule at the end of the list of existing icon rules, so that they read:

```
ic {
   existing-icon-rules ;
   compress { ti =Squash; pi = compress.px }
  }
```

Affecting a Group of Icons

It might be useful if files compressed with the UNIX System V **compress** utility are given the same icon as the utility itself or a related one if you prefer to design a new icon.

We can do this very simply by making use of the facility to define groups of files in the rule files. The UNIX **compress** utility gives the compressed versions of files the suffix *.Z* after their filenames. The rule file can assign an icon to all such files by giving the file specification ***.Z** where * is a wildcard matching any filename.

The icons affected can be restricted to just files or just directories by the suffix */F* or */D* respectively, so it would be better practice to give the file specification as:

```
*.Z / F
```

The full rule then becomes:

```
ic{
   *.Z / F { pi = ixi/icons/compressed.px }
  }
```

Files called *filename.Z* now automatically display the appropriate icon to identify them.

Building Intelligence into Your File Icons

The rule files also allow you to specify an action to be carried out when an icon, or group of icons, is triggered. Usually, triggering means double-clicking with one of the mouse buttons. In this example, we define a rule that automatically uncompresses a file that is in compressed format if it is double-clicked with the left-hand mouse button.

Let's assume that all compressed files have names with the suffix *.Z* as described above, and that the **uncompress** program is available. By a simple extension of this example, you could configure your system so that every text document or data file invokes the appropriate program tool when double-clicked, according to its filename.

Using Mouse Triggers

The standard mouse triggers are double-clicks with one of the mouse buttons. To make the Desktop Manager as portable as possible these triggers are normally pre-defined with names as follows:

Table 6.2.

Standard Mouse Triggers

Trigger Name	Function
s1	Double-click on mouse button 1 (the left button).
s2	Double-click on mouse button 2 (the center button).
s3	Double-click on mouse button 3 (the right button).

The definitions of s1, s2 and s3 are given in the Desktop Manager defaults file, and they can be altered to provide alternate ways of producing the three triggers on systems with fewer than three mouse buttons.

Writing Trigger Rules

The action to be performed when an icon is double-clicked is defined in the rules file by a trigger-action rule:

ta: *trigger-id* { *action-list* }

where the label `ta` identifies the clause as a trigger action. The *trigger-id* specifies one of the pre-defined mouse triggers, as set in the defaults file. Here we will use s1, a double-click with the left-most mouse button. The *action-list* specifies the commands that are actually run when the specified *trigger-id* is applied to an icon in the specified file-spec. Each command in the *action-list* has a prefix indicating whether it should be run either by the standard UNIX system shell, or by Xhibit. Here the action is to run the program **uncompress** with the file to create a new file of the same name, but without the .Z suffix. The full rule is:

```
ic{
   *.Z/F { ta : s1
          { ac
            {
              b : uncompress <%P0 >%D0/'basename %B0 .Z' ;
                    d : ddw %D0
                  }
            }
          }
}
```

This rule illustrates substitutions, which can be used in rules to refer to the components of the names of the files they apply to. The following substitutions are used:

Table 6.3.

Rule Substitutions

Substitution	Definition
%P0	The absolute pathname of the file.
%B0	The basename of the file.
%D0	The dirname of the file.

See "Rule Files" in Chapter 7, "Desktop Manager Reference," for definitions of these terms.

The first command in the action list uncompresses the double-clicked file to a file with the same name but without a *.Z* suffix. It constructs this name out of the file's dirname and basename, using the UNIX system **basename** command to return a filename with the *.Z* stripped off.

The second command in the action list runs the Desktop Manager command **ddw** to redraw the directory window to show the changed file icons.

This rule can be combined with the rule in the previous tutorial defining the appearance of *.Z* files. It should be included in the file *.xdtdirinfo* in the directory containing the compressed files.

These files then automatically run the **uncompress** utility, and they create an uncompressed version of the file in the same directory when they are double-clicked with the left-most mouse button.

Loading Files into a Program by Dragging

Another convenient way of telling the Desktop Manager to run a program with a file as data is to drag the data file's icon onto the program's icon. This action can also be specified in the rule files by making use of the drag triggers d1, d2, and d3. These triggers are sent to an icon when another icon is dragged onto it, and dropped, with the corresponding mouse button.

Because potentially any type of file could be compressed, it is inappropriate to use double-clicking to activate the **compress** utility. However, for this task we can make use of the dynamic triggers. In the next example we define a rule so that any file dragged onto the **compress** program with the left-most mouse button is automatically compressed to create a *.Z* file.

The full rule is:

```
ic
        {compress/FX    { ta : d1
                        {  ac
                            {
                                b : %P0 <%P1 >%P1.Z ;
                                d : chk %P1.Z
                            }
                        }
                    }
}
```

Here we limit the action to an executable file called *compress* by giving the suffix */FX*. The pathname of the dragged file is substituted for %P1, and the first command in the action list runs **compress** (%P0) on this to create a file of the same name with a *.Z* suffix. The Xhibit command **chk** then updates the new icon.

This rule should be included in the *.xdtdirinfo* file of the directory containing the **compress** program.

Building Intelligence into Directories

It is often useful to organize files into directories on a functional basis, such that all files of one type are kept together in one directory. The rule files conveniently allow actions to be performed when files are dragged into a directory window, making it possible to give certain directories in the filing system special attributes.

In the next part of the tutorial, we create a special directory called *compact*. Files dragged into this directory are automatically compressed, and the original version of the file is deleted to save the user's file space.

Drop Rules

The action of dropping an icon into a directory is defined by the drop rules. These have a similar form to the rules defining the action of double-clicking icons, except that there is no file specification. To create a drop rule specific to one directory, you have to put the rule file, named *.xdtdirinfo*, into the directory. Drop rules in user or system rule files apply to all directories.

The drop rule has the format:

dd { [td : *dynamic-trigger-id* { *action-list* }] I }

where the *dynamic-trigger-id* specifies one of the pre-defined mouse triggers. Here we use d1, a drag using the left-most mouse button.

As before, the *action-list* specifies the commands to be carried out when the icon is dropped into the directory window. In this case, the action is to compress on the file dropped, whose pathname is given by %P1. Then we wish to move the resulting compressed file into the directory and delete the original file:

```
dd{
  td : d1          {
                    d : mvi %P0 %P1 ;
            b : compress %P0/%B1 ;
                    d : ddw %P0
                   }
}
```

In this rule, we make use of the fact that %P1 contains the file's pathname, and %P0 the pathname of the directory into which it was dropped.

The first command in the action list moves the file into the directory and redisplays its icon there. The second command compresses it, using the directory pathname %P0 to refer to the file in its new location. Finally, the **ddw** command redraws the directory window to show the file's new icon.

This rule should be included in the *.xdtdirinfo* file in directory *compact*, which is to have this special action.

An Extension

A final refinement is to have the same action take place if we simply drag a file icon on to the icon of directory *compact*, rather than into its directory *window*. We can achieve this by including an additional icon rule in the rule file:

```
ic {
        compact/D       { ta : d1
                          { ac
                    {
                                d : mvi %P0 %P1 ;
                                b : compress %P0/%B1 ;
                                d : ddw %P0
                        }
                          }
                }
}
```

Chapter 7
Desktop Manager Reference

This chapter explains how to specify the appearance and behavior of the file icons that are displayed in the Desktop window.

Rule Files

Rule files control the behavior and appearance of the icons that represent files in the Desktop Manager. For example, they enable you to associate a picture and special mouse click actions with specific files or groups of files and to define what to do when one icon is dropped onto another. Rule files are text files, and they can be edited with a normal text editor. Special configuration tools may also be available to help with this task.

Components of Rule Files

A rule file consists of a number of components that can occur in any order, but each component should only occur once in any one rule file. The different components in a rule file are:

- **Icon rules.** They describe what the icon for each file looks like, which picture file is used to display it, what its title should be, and what should happen when the icon is triggered by double-clicking it or dragging another icon onto it.
- **Drop rules.** They describe what happens when icons are picked up and dropped onto the background of a directory window.
- **Desktop layout.** This lists which icons are out on the desktop and their positions. It is normally generated automatically.
- **Locked files.** Files that are locked out of the desktop and cannot be put back.

The Rule Hierarchy

There are four kinds of rule files, and they generally have the following precedence:

- **Local rule files.** They can be found in any directory, and provide rules that apply only to the files in that directory. They all have the name *.xdtdirinfo*.

- **Environment rule files.** They contain both rules and a list of files that are out on the desktop. Environments allow users to have different rules for different circumstances. For example, a user might have a programming environment and a text editing environment. At any time, each user has one active environment rule file, known as the "current environment." When a user changes environments, the icons currently on the desktop are saved in the old environment rule file and put away, after which a new set is read from the new environment rule file and placed on the desktop. Environment rule files can have any name, though it is suggested that they should end in *.xde*. The initial environment rule file for a user is specified through the X defaults mechanism.

- **User rule files.** Each user can have a user rule file called *.xdtuserinfo* stored in the user's home directory. It applies only to that user.

- **System rule file.** This file applies to all desktops running on a given machine. There is only one such file: */usr/lib/X11/xdt/xdtsysinfo*. System and user rule files are preloaded; therefore, any changes made to these only take effect when Xhibit is next run.

Writing Rule Files

Rule files are pure text files, and so they can be created and edited using any suitable program editor such as **vi** or **xedit**. In general, the layout is not critical, and spaces or new lines can be inserted to improve readability and to make the structure of the rules clearer. For example, the following two rule files are equivalent:

```
ic{*/D{pi=dir.px;}*/F{pi=file.px;}}
```

and

```
ic{
        */D{
                pi = dir.px ;
          }
        */F{
         pi = file.px;
          }
}
```

In general, spaces and new lines should be used to clarify the layout and function of rule files.

Special Characters

The following four characters have special meanings in rule files:

Character	Represents
%	percent
{	open brace
}	close brace
;	semicolon

Braces group together similar items, semicolon terminates other items, and percent introduces special phrases and instructions (the percent character was chosen to be distinct from the characters used for this purpose in the UNIX operating system). When a semicolon is followed by a close brace, the semicolon may be omitted.

There are a few places, such as when an icon title is specified, where spaces are significant. For example, the three spaces in the title in the following rule are not ignored:

```
ic { * { ti =Title for all icons; }}
```

If necessary, new lines and spaces can be included for formatting purposes in the icon title by surrounding them with a pair of % signs, and they can then be ignored. So the following rule is identical to the previous example:

```
ic { * { ti =Title %
    %for all icons; } }
```

Escape Sequences

The four special characters can be included in rule files, such as in the title of a filename, by preceding them with a % sign. Other characters can be included by preceding them with the escape sequence:

```
%!
```

Control Characters

You can include a control code, or other special character, in a rule file using the following sequence:

```
%# decimal-code #
%#0 octal-code #
%#0x (or %#0X) hexadecimal-code #
```

For example, the character double quote, character code 34, can be written as:

```
%#34#
%#042#          (34 decimal = 42 octal)
%#0x22#         (34 decimal = 22 hexadecimal)
%#0X22#
```

Comments

Comments can be included in a rule file by prefixing them with the sequence %//, which causes all characters up to the end of the line to be ignored. Long comments can be preceded by the sequence %/* and followed by the sequence %*/, which causes all characters between these two to be ignored. Note that comments introduced with the sequence %/* should obviously not contain the character sequence %*/, as this would indicate the end of the comment.

Referring to Filenames

When a file is referred to, its name may be used in four ways. These four ways have special (UNIX system) names:

Absolute pathname. This is the full name of the file, and it always begins with a slash.

Basename. This is the name of the file within its directory. It is the part of the absolute pathname following the last slash.

Dirname. This is the name of the directory holding the file. It is the part of the absolute pathname preceding the last slash.

Relative pathname If the file is in the desktop working directory or one of its subdirectories, the relative pathname is the absolute pathname without the name of the working directory or the slash that follows it. Otherwise, it is the same as the absolute pathname. For example, the various names of the file */user/fred/work/letter* are:

Absolute pathname:	*/user/fred/work/letter*
Basename:	*letter*
Dirname:	*/user/fred/work*

The relative pathname depends on the working directory:

Table 7.1.

Pathnames

Working Directory	Relative Path Name
/	*user/fred/work/letter*
/user	*fred/work/letter*
/user/fred	*work/letter*
/user/fred/work	*letter*
any other	*/user/fred/work/letter*

Note that there is one special case. The dirname of / is /. (slash-dot) and its basename is / (slash).

Components of Rule Files

This section describes the main components common to both icon rules and drop rules.

Triggers

To give the Desktop Manager the flexibility to work with any type of mouse with one to five buttons, the rules are usually expressed in terms of triggers rather than physical buttons.

Each trigger is labeled with a trigger-id, which is the letter s or d followed by a number or * that represents any mouse button.

Each trigger-id is assigned to a particular sequence of clicks, or presses, of the mouse buttons according to the *.Xdefaults* file.

The letter s is used for *static* triggers, where the mouse is not moved (e.g. double clicks), while d is used for *dynamic* triggers, or drags (e.g. dropping one icon onto another). For example, the usual assignment for a three button mouse is for the three trigger-ids s1, s2, and s3 to be double-left, double-center, and double-right click.

Another user, with only two buttons on the mouse, could set them to double-left, double-right, and double-both. Most users do not change the trigger to trigger-id mapping, because this is optimized to the particular mouse supplied with the computer system.

Action Lists

Action lists specify the commands that the Desktop Manager runs when an icon or directory window is triggered. The syntax for an action list item is:

{ac { [control : command ;] I } }

An action list can be empty, or contain one or more commands separated by semi-colons. Each command is prefixed by a control letter, specifying how the command is to be run. These prefixes are explained in the following table.

Table 7.2.

Action List Control Letters

Control Letter	Action
b	Command should be executed by the standard UNIX system shell */bin/sh*.
t	Command should be executed by the standard UNIX system shell within the standard terminal emulator (normally **xterm**).
d	The command should be executed by the Desktop Manager. The command executes directly rather than via the shell. This is more efficient, but it means that shell operations such as "<" and ">" are not available. For more information, see "Desktop Command Language" later in this chapter.

Example:

```
ac{
        t : vi myfile.1                          ;
        b : troff<myfile.1 >myfile.1.trf ;
        d : chk myfile.1.trf
 }
```

The commands are executed in order, with each being carried out after the previous one has finished. However, while the Desktop Manager is waiting for one command to finish executing, it can be doing other things, and several action lists can be executing at the same time.

Substitutions

Substitutions allow rules to make use of UNIX system environment variables and the filenames that correspond to the icons triggered or dropped. The name of a file may be substituted into the title of its icon, the icon's picture file, and the names of any of the files involved can be substituted into commands in action lists.

You can substitute UNIX system environment variables by surrounding the variable name with dollar signs and preceding it all with a percent sign. For example:

`%$`*variable_name*`$`

This line in a rule file causes the value of the UNIX system environment variable to be substituted at that point in the rule.

Filename substitutions are made by placing a substitution sequence in the rule file at the appropriate point. A sequence consists of a percent sign, a letter indicating the information to be substituted, and a number or asterisk. The number or asterisk determines the file or files for which information is to be substituted. The permitted cases are:

In icon titles:

0	The file of the icon.

In action lists for static icon triggers:

0	The file of the icon.

In action lists for dynamic icon triggers:

0	The file of the icon dropped on to.
1 - 9	The appropriate file in the list of those dropped on to the icon.
*	The files of each of the icons dropped in turn.

The values are separated by spaces. For example, if three icons are dropped, then %P* is the same as %P1 %P2 %P3.

In action lists for drop rules:

0	The directory of the directory window dropped into.
1	The file of one of the icons dropped into the window. For more information, see "Drop Rules" later in this chapter.

The letter can be any of the following:

Table 7.3.

Icon Filename Abbreviations

Abbreviation	Description
P	The absolute pathname of the file.
B	The basename of the file.
E	The unextended basename; anything after and including the last period (.) in the basename is omitted.
D	The dirname of the file.
R	The relative pathname of the file, except in the titles of icons within directory windows, when it is the basename of the file.
C	The class of the file (four capital letters). For more information, see "Classes" later in this chapter.
N	When followed by an asterisk, it gives the number of files dropped; for example, if three icons are dropped %N* is 3. Suppose that the local rule file for the directory */fred/jim* is:

```
ic
{
jane  { ta : s1 { ac {        b : %R0 -z    } } }
sarah { ta : d1 { ac {
                              b : mv %P* %D0;
                              b : echo %N* >%D2/count
          }       }   }
}
dd
{
          td : d1 { d : lni %P0/new %R1 }
}
```

And suppose that the current working directory is */fred,* then the following action lists are generated by the indicated actions:

Trigger-id s1 on /fred/jim/jane :

b : jim/jane -z

When several icons are dropped into an icon, the trigger-action rule is executed once with the full list of files dropped. For example, drop the icons */fred/one*, */jim/two*, and */ian/my/data* onto */fred/jim/sarah* with trigger-id d1:

```
b : mv /fred/one /jim/two /ian/my/data /fred/jim ;
b : echo 3 >/jim/count
```

When several icons are dropped into a directory window, the drop rule is executed once for each file dropped. For example, drop the same icons into the directory window for */fred/jim* with trigger-id d1:

```
d : lni /fred/jim/new one
d : lni /fred/jim/new /jim/two
d : lni /fred/jim/new /ian/my/data
```

Icon Rules

Icon rules describe the behavior and appearance of files when represented by icons. They include:

- the picture file used for the icon,
- the text for the icon's title,
- the action when the icon is triggered (double-clicked), and
- the action when another icon is dragged onto the icon.

The syntax for specifying an icon rule is:

ic { [*file-spec* { *file-rules* }] I }

The *file-spec* specifies a file, or group of files, to which the *file-rules* apply. It is usually clearer to group all the rules that apply to one class of files together.

The File Specification

The file specification restricts the files to which a clause applies—the "ruled files." The syntax for file specification is:

filename or: *filename* / *class*

where *filename* is an ambiguous name; see the following section. If no class is specified the rule applies to all classes.

Ambiguous Names

Ambiguous names are filenames that optionally contain certain special characters or wildcards. The following characters can be used in ambiguous names:

Table 7.4.

Wildcard Characters

Character	Represents
?	Any character. For example, *a?c* includes the files *abc* and *aac*, but not the file *abbc*.
*	Any sequence of characters, including none. For example, *a*c* includes the files *ac*, *abc*, *acbc* , and *as4hx..o6f,s:c*.
[abc]	Any of the specified set of characters. The set of characters can be abbreviated using a minus sign (-) to represent a range. For example, A-D is equivalent to ABCD. Ranges should only be between two letters of the same case, or two digits. Prefixing the set with carat (^) means not any character specified. For example, [^A-Za-Z]* means any file beginning with a character other than a letter.

NOTE: The special filename / may appear as a file-spec by writing it as //d (files called slash are directories).

Classes

Classes represent the properties of files in a concise form. These properties fall into four sets, and a file has exactly one property of each set. The properties are each represented by a letter (of either case), so that the class of a file consists of exactly four letters. The properties are file types, execute permissions, read/write permissions, and ownership. The letters used are as follows:

Table 7.5.

File Type

Abbreviation	Description
B	Block special file.
C	Character special file.
D	Directory.
F	Regular file.
G	Ghost (nonexistent) file.
I	Inaccessible file.
P	Pipe.
S	Symbolic link to nonexistent file; on machines without symbolic links no files have this type.

Table 7.6.

Execute Permissions

Abbreviation	Description
X	The user can execute the file.
A	The file has execute permission for someone, but not for the user.
N	The file does not have execute permission for anyone.

Table 7.7.

Read/Write Permissions

Abbreviation	Description
W	The user can read and write the file.
V	The user can read but not write the file, and the file has write permission for someone.
K	The user can read the file, and the file does not have write permission for anyone.
H	The user can not read the file.

Table 7.8.

Ownership

Abbreviation	Description
M	The user owns the file.
O	The user does not own the file.

NOTE: The codes G, I, and S imply the codes H, N, and O.

The term "class," in general, refers to a set of options, and it is described by a string of the appropriate letters. The order of the letters is not significant, and redundant letters are ignored.

For each set of properties, the letters of that set that appear should be viewed as being separated by the word "or," with the groups of letters from different sets being separated by "and."

For example, class BCNWVO is T(B or C) and N and (W or V) and OU.

If no letters of a class appear, then the class is read as if they all appeared. For example, class D is the same as DXANWVKHMO (both mean "all directories"). A number of extra codes can also be used in classes. These each stand for a common combination of the standard codes:

Table 7.9.

Class Descriptions

Class	Contents of Class	Class Description
Q	G, I, or S	Not a real file.
E	X or A	Executable by somebody.
U	A or N	Not executable by the user.
L	V or K	Readable but not writable by the user.
R	W or L	Readable by the user.

For example, DEO and DAXO have the same meaning. The following classes are the most useful in rule files:

Table 7.10.

Commonly Used Classes

Class	Description
D	Directories.
F	Files.
FE	Executable files.
FX	Files executable by the owner.
FN	Data files.
FNW	Data files that the owner can alter.
FNR	Data files that the owner can read.

File Rules

The file-rules consist of a sequence of at most one picture specification, at most one title specification, and any number of trigger actions. If the picture specification or title specification are not the last item in the list, they should be followed by a semicolon.

Picture Specification

The file specification assigns a picture file to the specified files. Picture specification syntax is:

pi = *picture-file*

For example:

* / D { pi = dir.px }

means that all directories are to use the picture in the picture file *dir.px.*

Title Specification

The title specification assigns a title to the specified files. Title specification syntax is:

ti = title

All spaces are significant between the equals sign and the semicolon or closing bracket. For example, the following clause sets the title of the **xcalc** program:

xcalc { ti =Calculator}

When using an ambiguous name, substitutions can be used to include the actual filename in the title.

Trigger Actions

Trigger action rules specify an action to be carried out when a specific trigger occurs with the mouse pointing to the icon. Trigger action syntax is:

ta : *trigger-id* { *action-list* }

The *action-list* specifies a list of commands to be carried out by the Desktop Manager or the operating system of the computer. For example, the following trigger rules say that trigger-id s1 on the appropriate icon causes the **xclock** program to be run:

```
ta : s1 { ac { b : xclock }}
```

Alternatively, the action list can be blank, in which case the trigger-id has no effect. For example, the following clause says that trigger s3 on a directory should do nothing:

```
* / D { ta : s3 { } }
```

The following, more complex example illustrates the use of trigger actions:

```
ic {
        *.c         { pi = csrc.px; }
        * / D       {
                pi = dir.px;
                ta : s1 { ac { d : ddw %P0 t }}
                    }
        [ab]*               { ti =AB file %B0; }
}
```

This has the following effect:

- All ending in *.c* use the picture found in the picture file *csrc.px.*
- All directories use the picture found in the picture file *dir.px.*
- When any directory is triggered with the trigger s1, then a directory window is opened to show that directory in time order.
- Any file beginning with the lowercase letters a or b has the icon title AB file followed by the basename of the file.

The rules take effect in the order in which they are specified, so the first rule in this example only affects files that have not already been given a picture by a previous rule.

Likewise, the second clause (pictures for directories) does not apply to directories whose names end with *.c* (though the third and fourth ones do).

Examples of Icon Rules

The following example defines icons and titles for the calculator, clock, and editor programs, and it causes the editor to be run when a file icon is dragged onto its icon, with that file loaded and ready for editing:

```
ic {
        xcalc   {       ti =Calculator; pi = xcalc.px  }
        xclock  {       ti =Clock;      pi = xclock.px }
        xedit   {       ti =XEdit;      pi = quill.px;
                        ta : s1 { }  ta : s2 { }  ta : d2 { }
                        ta : d1 { ac { b : %P0 %P1 } }
                        }
 }
```

The next example defines a rule that moves any file or files dropped onto a directory icon with the left-most mouse button:

```
ic {
        */D     {
              ta : d1{ ac {d : mvi %P0 %P*}}
                }
 }
```

The final example shows the standard definition of the icon rules for the Waste icon. The Waste icon is implemented as a directory with a suitable picture and title. For convenience, dragging with the left-most mouse button moves a file to the Waste directory, rather than copying it as is the usual default. The Waste directory can be emptied by double-clicking with the third mouse button.The command **rm -rf %P0/*** deletes all files in the directory.

```
ic {
        waste /d {
                  ti =Waste;
                  pi =waste.px;
                  ta : d1 { ac { d : mvi %P0 %P* } }
                  ta : d2 { ac { d : mvi %P0 %P* } }
                  ta : s3 { ac { b : rm -rf %P0/* } }
                  }
 }
```

Drop Rules

Drop rules describe the effects of dropping icons into directory windows. Drop rule syntax is:

dd { [td : *dynamic-trigger-id* { *action-list* }] I }

Drop rules in local rule files apply to the directory window of the directory holding the rule file. Drop rules in other rule files apply to all directories. Drop rules consist of a set of action lists, each associated with a dynamic trigger-id. As with icon rules, the first match is used. For example, the following drop rules cause dragging an icon into a directory window to copy or move the file, depending on whether mouse button 1 or 2 is used. This is usually the behavior of the Desktop Manager. In each case, %P1 is replaced by the pathname of the file dropped, and %P0 is replaced by the pathname of the directory window into which it was dropped.

```
dd{
        td : d1 { d : cpi %P0 %P1 }
        td : d2 { d : mvi %P0 %P1 }
 }
```

Multiple Drops

An important difference to note between drop rules and icon triggers is the action taken when several icons are dropped into a directory window at the same time. The action list is duplicated several times, one copy for each icon dropped, and then each copy is modified by the substitution system to include details about that icon.

For example, suppose we have the action list:

```
{
 d : mvi /waste %P1          ;
 b : echo %P1 >>/waste/.filelist
}
```

We drop the three icons */fred/fred*, */fred/jim*, and */fred/sheila*. Because %P1 is replaced by the pathname of the icon dropped, the action list actually executed is:

```
{
 %// Spaces have been added to the commands in
 %// order to make their meaning clearer.
 d : mvi  /waste /fred/fred ;
 b : echo         /fred/fred  >>/waste/.filelist ;
 d : mvi  /waste /fred/jim  ;
 b : echo         /fred/jim   >>/waste/.filelist ;
 d : mvi  /waste /fred/sheila        ;
 b : echo         /fred/sheila >>/waste/.filelist
}
```

Desktop Layout

The desktop layout list describes the files that are on the desktop, together with their positions. It is normally generated automatically by the Desktop Manager, but it could be modified by a program to alter the initial appearance and layout of a desktop. The desktop layout list is ignored if it is not in an environment file. Layout syntax is:

dt { [*filename* [@ *position*] ;] I }

The *position*, if present, consists of one of the following:

G followed by the coordinates in tidying grid units,

P followed by the coordinates in pixels, or

F indicating the icon is to be placed at the first free position of the grid.

Omitting the position code is the same as specifying a code of F. For example:

```
dt {
        /                @ G  0,  0 ;
        /usr/bin @ G  1,  0 ;
        /fred            @ F      ;
        /fred/main       @ G  4,  7 ;
        /bin             @ P 211, 874 ;
        /fred/data
  }
```

Locked Files List

The locked files list allows icons to be locked on the desktop. Locked file syntax is:

```
lf { [filename ; ] I }
```

Locked files lists in the system and user rule files apply whenever the Desktop Manager is run. A locked files list in an environment file applies only while that environment file is current. Locked files lists are ignored in local rule files. For example:

```
lf {
        /     ;
        /bin  ;
        /usr/bin
  }
```

Mapping Triggers

For easier portability, the Desktop Manager converts clicks on the mouse buttons first into triggers, and then into trigger-ids. This mechanism is controlled by four items in the X defaults mechanism; these are the mapping (a string), the maximum motion (a number of pixels), a threshold down time, and a maximum up time (both times measured in milliseconds). The trigger mapping setup when your system is supplied is normally optimum for your mouse and configuration, but you can modify the actions to suit your own requirements. The conversion is done in two stages. First, the motions and button presses are converted into triggers. Second, the triggers are converted to trigger-ids through the mapping string.

Triggers

A trigger is a set of closely spaced button presses and releases. The easiest way to think of a trigger is as a series of "steps." Each step starts when, with all the mouse buttons up, one of the buttons is pressed. It ends the next time all the buttons are up.

Trigger Steps

A step is labeled by giving the numbers of all the mouse buttons that are depressed at any time during the step, no matter what order they are in, or how long they are down. For example, all three of the following examples would be labeled as 1 and 3, or 13 for short:

- Press button 1, press button 3, release button 1, and release button 3.
- Press button 3, press button 1, release button 1, and release button 3.
- Press button 1, press button 3, release button 3, press button 3, release button 3, and release button 1.

NOTE: "press" means press and hold down the button.

The three types of step are defined as follows:

Table 7.11.

Trigger Steps

Step	Mouse Movement	Interval Between Events
short click	< maximum motion	< threshold down time
long click	< maximum motion	> threshold down time
drag	> maximum motion	—

A short click and a drag are described by giving the numbers of the mouse buttons: 13

A long click is described by giving the numbers of the mouse buttons followed by a plus sign: 13+.

Triggers

A trigger is a sequence of steps, and is described by giving the steps, separated by commas. For example, the trigger double-click on button 2U is described as 2,2. If the last step in the sequence is a drag, the trigger is defined as a dynamic trigger, and the Desktop Manager signifies detection of the drag by changing the cursor to the drag or multi-drag cursor. Other triggers are defined as static triggers. A trigger ends when either no button is pressed for the maximum up time after a step, or at the end of a drag, whichever comes first. All triggers containing more than five steps are ignored by the Desktop Manager.

Converting Triggers to Trigger-Ids

All triggers that you want to be interpreted by the Desktop Manager must appear in the mapping string. This consists of a sequence of mappings, separated by semicolons (spaces anywhere in the mapping string are ignored). There are three things that can occur in the mapping string:

- Static trigger mappings
- Dynamic trigger mappings
- Macro definitions

Static Trigger Mappings

Each static trigger mapping maps a static trigger to a trigger-id. Static trigger mapping syntax is:

static-trigger = trigger-id

Static-trigger is a list of steps, separated by commas. Trigger-id is an s followed by a number. A static trigger can also be used to control the selection of icons. This is done by using one of the following codes instead of a trigger-id:

Table 7.12.

Trigger Mapping Codes

Code	Description
+s	If the current icon (the icon that is under the cursor) is not selected, then select it.
-s	If the current icon is selected, then deselect it.
!s	Deselect all selected icons, and then select the current icon.
~s	If the current icon is selected, then deselect it. Otherwise, select it.

For example, the following says that a short click on button 1 selects the current icon in addition to any icons already selected, while a long click selects it on its own. Either type of click on button 2 deselects the icon:

```
1=+s ;  1+=!s ;  2=-s ;  2+=-s
```

Dynamic Trigger Mappings

Each dynamic trigger mapping maps a dynamic trigger to a dynamic trigger-id. Dynamic trigger syntax is:

```
dynamic-trigger = trigger-id
```

Dynamic-trigger is a list of steps separated by commas. Trigger-id is a "d" followed by a number.

For example, the following says that a drag with button 2 on its own generates trigger-id d2, but if preceded by a short click on button 4, it generates trigger-id d6:

```
2=d2 ; 4,2=d6
```

Dynamic triggers cannot be used to control icon selection.

Macro Definitions

Macro definitions allow one or more buttons to be abbreviated to a single letter. This allows mappings to be made more abstract, and so easier to convert for a different number of buttons.

For example, suppose that you have designed a set of mappings for a three-button mouse, and that you want to convert it to work on a two button mouse. One way might be to say that the center button is represented by using both left and right buttons together. By specifying all the mappings in terms of the letters L, C, and R, rather than the numbers 1, 2, and 3, they are easier to change (especially as the right button changes from being number 3 to being number 2).

A macro definition consists of a set of button numbers, an equals sign, and then a single letter. That letter can then be used in any future trigger description or macro definition.

For example, a trigger mapping with three static trigger-ids, three dynamic trigger-ids, and three selection control triggers, might be written as follows for a three-button mouse:

```
1=L   ; 2=C   ; 3=R   ; 2
L=!s  ; C=+s  ; R=-s  ; 2
L,L=s1 ; C,C=s2 ; R,R=s3 ; 2
L=d1  ; C=d2  ; R=d3
```

The backslashes indicate that the mapping is continued on the next line. To convert to the two-button mouse, change the first line to:

```
1=L   ; 12=C  ; 2=R   ; 2
```

The mapping then becomes equivalent to:

```
1=!s  ; 12=+s  ; 2=-s  ;2
1,1=s1 ; 12,12=s2 ; 2,2=s3 ;2
1=d1  ; 12=d2  ; 2=d3
```

Desktop Command Language

The Desktop command language, DCL, allows certain actions to be carried out within the Desktop Manager. These actions are mainly concerned with icons, directory windows, and copying and moving files.

DCL functions can be piped into the Desktop Manager or used in rule files. The advantages of using DCL commands are that they automatically take care of updating the desktop, and they do not rely on the availability of particular UNIX binary files.

Commands in DCL consist of words separated by spaces. The end of a command is marked by the mechanism that initially generated the command. For example, within a rule file, the end of a command is indicated by a semicolon.

A backslash causes the following character to be part of the current word, even if it is a space character. For example, the following command contains only two words:

```
this\ is\ a\ command   with\ only\ two\ words
```

All valid commands begin with a word of three lowercase letters, followed in some cases by a number or arguments. Some commands require an exact number of arguments, and the effect of having the wrong number is undefined. Other commands accept any number of arguments.

The following notation is used for describing the arguments of commands:

file Represents a file name

dir Represents an existing directory.

Desktop Commands

Terminate execution - die
Syntax: **die**

Terminates the Desktop Manager.
Change environment - ndt
Syntax: **ndt** *file*

Changes the current Desktop Manager environment to the specified file.

Catalogue desktop - cdt
Syntax: **cdt** *file*

Writes a list of the icons on the desktop and their positions into the specified file, replacing any such list already in that file. The Desktop Manager's current environment is not changed.

New desktop -rdt
Syntax: **rdt** *file*

Switches to a new desktop environment without saving the old one.

Trigger action - act
Syntax: **act** *static-trigger-id file*

Executes the action list as if the specified trigger had been used on the specified file.

Note that in each case any commands following the **act** command in the action list are executed immediately, independently of the triggered action list.

Syntax: **act** *dynamic-trigger-id file* [*file*] . . .

Executes the action list as if the list of files had been dropped on the first file named. Note that this command is completed as soon as the first command in the list starts executing.

Drop action - drp
Syntax: **drp** *dynamic-trigger-id dir* [*file*]

Executes the action list as if the list of files had been dropped on the open window of the directory. The directory window does not need to be open. Note that this command is completed as soon as the first command in the list starts executing.

Open directory window - ddw
Syntax: **ddw** *dir* [*flags*]

Opens the directory window for the directory, if it is not already open, brings it to the front, and displays it in the format given by the flags. The following table lists the valid flag types:

Table 7.13.

Open Directory Window Flags

Flag	Meaning
i	Display by icon (default).
n	Display by name.
a	Sort alphabetically (default).
t	Sort by time.
c	Sort by class.
u	Sort in extra order.

This command can be used both for opening new windows and for altering the appearance of existing ones.

Replace directory contents - rdw
Syntax: **rdw** *dir1 dir2* [*flags*]

If the directory window for directory *dir1* is open, its contents are replaced by directory *dir2*. The flags have the same meanings as for the **ddw** command. If there is already an open window for directory *dir2*, it is brought to the front, and the window for directory *dir1* is closed.

Close directory window - cdw
Syntax: **cdw** *dir*

Closes the specified directory window if it is open.

Bring window to front - btf
Syntax: **btf** *dir*

Brings the specified directory window to the front if it is open.

Get out icon - goi
Syntax: **goi** *file* [*position*]

Places the icon of the file on the desktop. If a position is specified, it should be one of the following forms:

Px,y	position in an exact number of pixels
Gx,y	position in the standard tidying grid
F	first free position on the grid

where x and y are numbers.

Put back icon - pbi
Syntax: **pbi** *file*

Puts back the icons of any of the specified files that are on the desktop (except locked ones).

Tidy desktop - tdf
Syntax: **tdf**

Tidies the desktop.

Reorganize desktop - tds
Syntax: **tds**

Reorganizes the desktop.

Copy file - cpi
Syntax: **cpi** *dir file*

Copies the specified files into the specified directory. If the directory window is open, new icons appear in the window.

Link file - lni
Syntax: **lni dir file**

Links the files into the specified directory. If the directory window is open, new icons appear in the window.

Move file - mvi
Syntax: mvi dir file I

Moves the files into the specified directory. If the directory window is open, new icons appear in the window. If the icons of the specified files are on the desktop, their titles change if necessary. If the icons of the specified files are visible in directory windows, they disappear.

Update icons - chk
Syntax: **chk** [**-R**] *file*

Ensures that any icons visible for the specified files have the correct appearance, even if the properties of the file, or any of the applicable rule files, have changed since the icon was first made visible.

Options:

-R Removes the icon from the desktop and directories if the file does not exist.

Picture Files

The Desktop Manager icon pictures, background patterns, and control patterns are held in picture files. These can be edited using a bitmap editor.

You can find picture files in */usr/include/X11/bitmaps*, which contains several subdirectories:

Table 7.14.

Picture Files

File	Context
ixi_cursors	Cursors used by the Desktop Manager.
ixi_icons	Icon pictures used by the Desktop Manager.
ixi_keys	Desktop Manager window-managing control boxes.
ixi_logos	Company logos.
ixi_misc	Large bitmaps for Desktop Manager warning boxes.
ixi_textures	Background pixmaps for the Desktop Manager.

When the name of a picture file begins with a slash, the file can be found without help. The picture directory (looked up in the X defaults mechanism) is used by the Desktop Manager to find picture files whose names do not begin with a slash.

If the name of the picture file does not begin with a slash, then it is looked up in two places. First, the name of the picture directory, and a slash, are prefixed to the name of the file. If this file is not found, or if there is no picture directory item in the X defaults, then the standard prefix */usr/include/X11/bitmaps* is used instead.

Suppose that the picture directory is set to */user/fred/pictures* and we are trying to find the picture file *core.pic*. Then the Desktop Manager looks for these files:

/user/fred/pictures/core.pic
/usr/include/X11/bitmaps/core.pic

It is permissible for the picture file to have a slash in its name, so that *patterns/checked.pic* would be looked for in:

/user/fred/pictures/patterns/checked.pic
/usr/include/X11/bitmaps/patterns/checked.pic

Format of Picture Files

Picture files are an extended form of X bitmap files, and X bitmap files are, therefore, always legal picture files. Picture files can also be generated with the **pixmap2c** utility (if available on your system). A picture file consists of two kinds of items: configuration items and data items. The order of individual items is not constrained except that all configuration items must occur before all data items.

Configuration Items

There are six kinds of configuration items.

Each item must be on a separate line, and consists of the prefix **#define** followed by a name and a value, with spaces or tabs separating each of the three parts. The three parts of the item must all occur on the same line, and the pound sign (#) must be in the first column.

The first part of each item name should correspond to the first part of the name of the picture file, containing only the characters [A-Za-z0-9_]. In the following examples the items are given for a picture file *pic.px*:

pic_width
pic_height

These two items must occur. Their values are numbers, and give the width and height of the picture. If the picture is used for an icon, button, or cursor, this is the size of the object. If it is used as a background, the picture is tiled across the area; these items are still required to enable the data items to be interpreted.

pic_x_hot
pic_y_hot

These two items must both occur or both be omitted. They are only used if the picture is the data portion of a cursor, and indicate the coordinates within the picture where the cursor is actually located.

For example, if both values are zero, the actual point of the cursor would be the top left corner of the picture. If the value is -1, both must be -1 and it is treated as if the entire item was omitted.

pic_fg
pic_bg

These two items are optional. Their values must be names of colors, surrounded by double quotes, giving the foreground and background colors of the picture. If the color does not begin with a hash sign, then its meaning depends on your X server. If it does begin with a hash sign, then the remainder of the color name encodes the actual color.

Your implementation of the X system may interpret spaces in a name. Spaces are not permitted in the encoding format.

The encoding gives the red, green, and blue components of the color, in that order, as one, two, three, or four hexadecimal digits each. Components written 5, 50, 500, and 5000 are all the same, and differ from 05, 050, and 0005. Refer to the X(1) manual page for more information. If these items are omitted, then the foreground is black and the background is white.

For example, black is "#000000000000" or "#000," white is "#ffffffffffff," and red is "#ffff00000000."

Data Items

There is one kind of data item—the picture data. It consists of the sequence

static unsigned char *pic_bits* [] = { *data* }

where **unsigned** can be omitted and *data* represents the actual data, consisting of a sequence of two- digit hexadecimal values, each prefixed with 0x and separated by commas.

There may be up to 20 such values per line, though it is usually 12.

If the width and height of the picture are W and H, respectively, there should be a total of ((W+7)/8)*H values, (W+7)/8 for each row of the picture (the division is rounded down, rather than being an exact number). Each value represents eight consecutive pixels, except that the last value in the row can represent less.

Example

The following example shows a sample picture file:

```
#define menu_d_width 16
#define menu_d_height 16
#define menu_d_x_hot 14
#define menu_d_y_hot 5
#define menu_d_bg "black"
#define menu_d_fg "white"
static char menu_m_bits[] = {
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0e, 0x00,
        0xfa, 0x1f, 0x02, 0x20, 0xc2, 0x1f, 0x02, 0x02,
        0xc2, 0x03, 0x02, 0x02, 0xfa, 0x01, 0xfe, 0x00,
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
```

Defaults Files

The X defaults mechanism is used by many X utilities to obtain information about which options they are to use. In particular, it is used by the Desktop Manager for a range of information.

The X defaults mechanism works by reading a number of files and constructing a database from them. The mechanism is described in Chapters 1 through 4 of this guide. The database used by the Desktop Manager is built out of the following sources (in their order of precedence):

The command line.
The file named in *$XENVIRONMENT*.
The X server's RESOURCE_MANAGER property (loaded by **xrdb**).
The file *${XAPPLRESDIR} Xhibit*.
The file */usr/lib/X11/app-defaults/Xhibit*.

$XENVIRONMENT is the standard UNIX environment variable. If the file *$XENVIRONMENT* does not exist, then it is replaced by:

$HOME/.Xdefaults-machine

where *machine* is the name of the machine that the Desktop Manager is running on, and *$HOME* is the standard UNIX environment variable.

If the X server's RESOURCE_MANAGER property is undefined, the file *$HOME/.Xdefaults* is used instead. If *$XAPPLRESDIR* is undefined, *$HOME/Xhibit* is used.

If this, or any of the other files does not exist, it is skipped (so that none of the files are necessary). If the database does not contain any entries matching a particular item, it uses built-in defaults.

Defaults Items

Each item is listed in the form:

name name

class class

The following pair is prefixed to each item before it is looked up:

xhibit

Xhibit

Objects that are options should have values that are on or off. The words that are understood by the Desktop Manager are given in the section titled "Message Files."

Text

`font`

`Font`

This specifies the name of the font that is used by the Desktop Manager for text; there is a default font.

`textMargin`

`TextMargin` `(number:2)`

This specifies the amount of space that should appear around all text displayed by the Desktop Manager.

Icon Layout

When the reorganize option is used, the icons can be spread out in rows or columns.

```
iconGrid   horizontal

IconGrid   Horizontal   (option:off)
```

The default specifies that the icons should be spread out in columns.

```
iconGrid        spacing        x

IconGrid        Spacing        X        (number:120)
```

This specifies the number of pixels apart that icons should be arranged horizontally on the desktop when it is tidied. This distance is measured from the center of each icon.

```
iconGrid   spacing   y

IconGrid   Spacing   Y   (number:40)
```

This specifies the number of pixels apart that icons should be arranged vertically on the desktop when it is tidied. This distance is measured from the center of each icon.

```
directory   aisleWidth

Directory   AisleWidth   (number:8)
```

This is the minimum number of pixels that should be left between each icon in a directory window when it is first opened and whenever it is tidied.

File Defaults

```
initialEnvironmentRuleFile
InitialEnvironmentRuleFile      (filename:xdtinitial.xde)
```

This is the name of the initial environment rule file.

```
isWindowManager
IsWindowManager   (option:off)
```

This option determines whether the Desktop Manager runs as a window manager (option:on) or as an ordinary program (option:off).

```
pictureDirectory
PictureDirectory   (filename:no default)
```

See "Picture Files" earlier in this chapter for details on the meaning of this value, which should be the name of a directory and should begin with a slash.

Note that if the Desktop Manager cannot find a picture file in *PictureDirectory* it looks in */usr/include/X11/bitmaps*.

Triggers

The following values are used to convert triggers to trigger-ids; see "Converting Triggers to Trigger-ids," earlier in this chapter.

```
triggers   mapping

Triggers   Mapping   (string)
```

The default mapping string (spaced out on several lines) is:

```
1=!s  ; 2=+s   ; 3=-s   ; 4=~s   ;
1,1=s1 ; 2,2=s2  ; 3,3=s3  ; 4,4=s4  ; 5,5=s5 ;
1=d1  ; 2=d2   ; 3=d3   ; 4=d4   ; 5=d5
```

The remaining items are numbers that alter the thresholds used in the conversion. The details of the conversion mechanism are described here.

```
triggers   maxMotion
Triggers   MaxMotion              number:3 pixels)
triggers   maxUpTime
Triggers   Time                   (number:700 ms)
triggers   thresholdDownTime
Triggers   Time                   (number:500 ms)
```

Cursor Shapes

Cursors are used by the Desktop Manager for the following functions:

Table 7.15.

Functions that Require a Cursor

Function	Description
busy	The Desktop Manager is doing something.
drag	An icon has been picked up and is being moved.
multiDrag	More than one icon has been picked up and is being moved.
idle	The Desktop Manager is waiting for a command.
menu	The pointer is over a menu.
none	A window is being moved or resized. This cursor should be blank.
alert	Alert window is being displayed.
fatal	Fatal window is being displayed.

The default cursors are built into the Desktop Manager, but any of them can be redefined. Each pair of pictures forms a cursor shape.

```
busy      data
Cursor    Bitmap      (picture-filename)

busy      mask
Cursor    Bitmap      (picture-filename)
```

and so on for the other cursor names.

Desktop Appearance

The following values give the geometry of the desktop window if the Desktop Manager is not running as a window manager.

```
geometry
Geometry      (X geometry specification)

desktop                  x
Desktop                  X                (number)

desktop                  y
Desktop                  Y                (number)

desktop                width
Desktop                Width              (number)

desktop               height
Desktop               Height             (number)
```

These numbers are supplied as preferences to the window manager but can be ignored. The background patterns used by the desktop are specified by:

```
desktop     backgroundPixmap
Desktop     BackgroundPixmap     (picture-filename)

directory   backgroundPixmap
Directory   BackgroundPixmap     (picture-filename)
```

```
directory    scrollbar    backgroundPixmap
Directory    Scrollbar    BackgroundPixmap    (picture-filename)
```

Each item should be the name of a picture file. This picture is tiled across the area in question. The default pictures are built into the Desktop Manager.

Menus

The following items specify the menu drop-shadow width, the height of an ordinary dividing bar, and the height of the bar beneath the menu title.

```
            itemBar        height
Menu        Bar            Height    (number:2)
            titleBar       height
Menu        Bar            Height    (number:5)
Menu        ShadowWidth               (number:2)
```

Message Windows

The following items affect the visible appearance of message windows:

```
            borderWidth
Message     BorderWidth      (number:4)
            innerMargin
Message     InnerMargin      (number:10)
```

Name Entry Box

The following items affect the visible appearance of name entry boxes:

```
rename   border
Rename   Border       (number:3)
rename   width
Rename   Width        (number:14)
rename   step
Rename   Step         (number:9)
```

Launching Programs

The following items control whether the cursor should be changed to the "launch" cursor when a program is run.

```
process   launch   show

Process   Launch   Show              (option:on)

process   launch   cursor   data

Process   Launch   Cursor   Bitmap   (cursor picture file)

process   launch   cursor   mask

Process   Launch   Cursor   Bitmap   (cursor picture file)

process   launch   time

Process   Launch   Time              (number:3)
```

If the show option is on, then the cursor is changed to the "launch" cursor for the time given (in seconds).

```
process   showBorder

Process   ShowBorder       (option:off)
```

This option indicates whether process window borders should be visible inside the process window frame.

The following items control whether process window and icon positions are remembered by the Desktop Manager or whether it puts the process icon under the close box of its window and opens a window above the process icon.

```
process   windowLock

Process   WindowIconLock      (option:off)

process   iconLock

Process   WindowIconLock      (option:on)
```

Buttons and Icons

The following items give the pictures for the buttons and icons needed to run the Desktop Manager:

```
directory   button_name   pixmap

Directory   Button        Pixmap    (picture-filename)
```

This is for each of the directory window buttons by icon, by name, close, and grow.

```
message   goAway   pixmap

Message   Button   Pixmap    (picture-filename)
```

This is for each of the messages alert, fatal, greeting, and information.

```
process   button_name   pixmap

Process   Button        Pixmap    (picture-filename)
```

This is for each of the process window buttons grow and close.

```
default   pixmap
Icon      Pixmap      (picture-filename)
```

This is the picture used where the rule files do not specify a picture for a file.

```
newFile   pixmap
Icon      Pixmap      (picture-filename)
```

This is the picture used while you are creating a new file. For example, for the **New empty file** menu option.

```
process   default   pixmap
Process   Icon      Pixmap    (picture-filename)
```

This is the picture used for a process icon.

Message Files and Language Support

All Desktop Manager messages are kept in a message file that can be edited by the user, making it very easy to use the Desktop Manager in a foreign language or tailor the Desktop Manager messages to specific requirements. By using the X/Open standard Native Language Support, the Desktop Manager adheres to a common method for provision of language information. If your computer supports the NLS system, then the Desktop Manager uses it. Otherwise, the Desktop Manager provides a similar mechanism itself. The differences between the two systems are described here.

NLS Systems

On NLS systems, the message file described in this section must be converted into a special format known as an NLS catalogue. This is done with the **gencat** utility. The search mechanism described here is then used by the Desktop Manager to find the catalogue, rather than the message file.

The format of message files accepted by **gencat** can be more complex than described here. If so, you can make use of any facilities supported on your system. See your system administrator for the specific location of your NLS catalogues.

Other Systems

On systems that do not support NLS, Xhibit uses the message file directly, without converting it to a different form. The search mechanism is used to find the message file.

Message files should only use the facilities described in this section. You may store message files anywhere on your computer, but we suggest that those intended for general use be stored in the directory */usr/lib/X11/xdt/xdtmessages*. The Desktop Manager looks for message files or catalogues in various places, depending on the values of two environment variables: LANG and NLSPATH.

The LANG Environment Variable

LANG is used to determine which of the Desktop Manager's message files you wish to use. The X/Open rules state that LANG should be in one of three forms:

language
language_territory
language_territory.codeset

where *language* gives the name of the language that you want your messages to be in, *territory* indicates territorial differences (for example, between UK and US English, or among French, Belgian, and Swiss usages in the French language), and *codeset* selects a particular character set. If any part is omitted, then a default should be used P. This default may vary from system to system. All names should be in English.

For example, a particular user might set LANG to french to indicate that they want messages in French, to french_swiss to further indicate that they wish Swiss conventions to be used (the default on their system might be Belgian), or to french_swiss.8859 to indicate that the message file written in the ISO character set IS8859/1 should be used.

The NLSPATH Environment Variable

NLSPATH should be set to a sequence of filenames, separated by colons. The Desktop Manager uses the first of these files that it finds. The percent character is used to indicate that something should be substituted:

Table 7.16.

NLS Abbreviations

Abbreviation	Description
%L	The value of LANG.
%l	The language element of LANG.
%t	The territory element of LANG, if specified.
%c	The code set element of LANG, if specified.
%N	The string **xdt.**
%%	The percent character.

For example, suppose that LANG is set to french_swiss and NLSPATH is set to the list %L:%N.cat/%l:/nls/%l/terr_%t/code_%c/prog_%N. Then the files looked for are:

```
french_swiss
xdt.cat/french
/nls/french/terr_swiss/code_/prog_xdt
```

The default values used by the Desktop Manager are:

```
LANG english
NLSPATH /usr/lib/X11/xdt/xdtmessages/%L
```

The Format of the Message File

The messages in the message file are divided into numbered sets, and each message is given a number within its set. This system allows related messages to be grouped together.

Each set of messages starts with a line consisting of the word $set followed by a space and the set number. Anything following the number is ignored. Each message then appears on a separate line, consisting of the message number followed by a space and the text of the message. The message sets and the messages within each set can be in any order, but each set must be all together. It is not possible to split up a set and have two $set lines with the same number.

Comments can be added to message files. A comment line starts with a dollar sign and then a space or a tab. Anything following the space or tab is then ignored. In addition, blank lines are ignored. Here is a simple example of a message file that contains two sets with a total of five messages.

```
$ This is an example message file.
$ These two lines are ignored.

$set 5 This is message set 5.
1 This is message 1 of set 5.
8 This is message 8 of set 5.
999 This is message 999 of set 5.
$set 26
86 This is message 86 of set 26.
4  This is message 4 of set 26.
```

There are a few special characters that can be added to messages. A backslash can be used to add non-printable characters:

Table 7.17.

Non-printable Characters

Character	Meaning
\n	new line
\t	tab
\b	backspace
\	backslash

A message can be continued on more than one line by ending the first line with a backslash.

Finally, certain messages have values, such as filenames, substituted in them. This is done by placing the string *%n*$s in the message, where *n* is a digit. For each substitutable object, the appropriate string should occur exactly once. If there are more than one, they may occur in any order. The messages that have strings substituted in them are given in the following table.

Table 7.18.

Message Substitutions

Set	Message	Meaning of `%1$s`	Meaning of `%2$s`	Meaning of `%3$s`
11	2	Internal information		
11	16	Internal information		
11	36	Type of error	X request name	Failed resource (in hex)
12	11	New name entered	File being duplicated	
12	12	New name entered		
12	13	New name entered		
12	14	Name of directory		
12	15	File to be copied	Directory copied into	
12	16	File to be moved	Directory moved into	
12	17	File to be linked	Directory linked into	
12	18	File being renamed		
12	20	File being duplicated		
12	21	File being copied	File copied to	
12	23	File being renamed		
12	24	File being written to		
12	31	File name		
12	32	Locus version #	IXI version #	
12	33	Locus version #	IXI version #	
12	34	File name		
12	36	File name		
12	37	File name		
12	43	File name		
12	44	Character	Line number	Filename
12	45	Line number	File name	
12	46	Line number		

Notes on Individual Messages

Set 1	message 11	Used when renaming files or entering new filenames.
	message 12	Used as the name of any process that has not specified a name.
	message 13	Used as the icon name of any process that has not specified an icon name.
	message 16	The name that the Desktop Manager is given if run under another window manager.
	message 17	The icon name that the Desktop Manager is given if run under another window manager.
Set 2	messages 1 and 2	The names of colors that are looked up in the server color database. Message 1 is used for black and message 2 for white.
Set 3	message 100	Should contain a number (say "p").
	messages 101 to 100+p	The strings that the taken by the defaults mechanism to mean "on."
	message 200	Should also contain a number (say "q").
	messages 201 to 200+q	The strings that are taken to mean "off."
Set 10	These messages are used to augment the messages in Set 11.	

Set 11	These messages are used when a fatal error occurs.	
	message 1	Used for all fatal errors that prevent the Desktop Manager from using graphics.
	message 2	Used for all fatal errors in *xdtforker*.
	messages 3 and 9	Used for all fatal errors that can be reported in a stop box.
Set 12	Used for warnings and other messages.	
	message 1	This message is placed in front of all messages displayed in a warning box.
	messages 32 and 33	Message 32 is used when the Desktop Manager is running as a window manager, and message 33 otherwise.
Set 21	These messages form the contents of the desktop menu.	
Set 22	These messages form the contents of the directory menu.	
Set 50	message 100	Should contain a number (for example, "p").
	messages 101 to 100 + p	The strings that refer to the right side of something.
	message 200	Should contain a number (for example, "q").
	messages 201 to 200 + q	The strings refer to the bottom of something.
	message 300	Should contain a number (for example, "r").
	messages 301 to 300 + r	The strings that refer to the left side of something.
	message 400	Should contain a number (for example, "s").

messages 401 to 400 + s	The string that refers to the top of something.

Command-Line Options

The following command-line options can be specified when the Desktop Manager is run. Where they conflict with settings and values supplied by the X defaults mechanism, the command-line options take precedence.

Table 7.19.

Command-Line Options

Option	Function
-display	The name of the display Desktop Manager should use.
-window	Runs the Desktop Manager in a window, rather than as a window manager.
-manager	Runs the Desktop Manager as a window manager, rather than in a window.
-font *font*	Specifies a font to be used.
-geometry geometry	(ignored if **-manager**) Specifies the size of the window to be used.
=geometry	equivalent to **-geometry**.
-xrm *resource*	Specifies a resource name and an optional value to add to the defaults database.

If both **-window** and **-manager** are specified, the one appearing closest to the end of the line is used.

The geometry option should take one of the following forms:

widthxheight+x+y. For example, 500x500+100+100

widthxheight

+x+y

Any dimensions missing are taken from the X defaults mechanism. Either plus sign may be replaced by a -P; the number is then the distance between the right or bottom edge of the Desktop Manager window and the corresponding edge of the screen.

X.Deskware Support Utilities

Several simple X11 support utilities are provided to help the non-programmer develop sophisticated interactive dialogues in icon action and shell scripts.

Table 7.20.

X11 Utilities

Utility	Meaning
fyi	For your information.
gti	Get text input.
yni	Yes or no input.
xdtinfo	File information.

These utilities accept standard X11 command-line options, such as **-d** (display), or **-fn** (font name). The summaries following only describe the extra options they use.

fyi -for your information

Syntax: fyi [*options ...*] *displaytext*

Options:

- **-override** Causes the window manager to ignore **fyi**.
- **-help** Prints **fyi** usage information on the standard error.

- **-picture** *picturefile*
  Specifies the file name of the picture to display.

- **-type** Specifies any string to be used as a title. For example: "Warning" or "Error."

gti - Get text input.

Syntax: gti [*options ...*] [*default_string*]

Options:

- **-prompt** *prompt* Prompts the user for information.

- **-length** *display_length* Specifies the length of the text entry.

- **-step** *display_step* Specifies how many characters the entered text should scroll past if the text becomes longer than the entry area.

yni - yes or no input

Syntax: yni [*options ...*] *text*

Options:

- **-override** Causes the window manager to ignore **yni**.

- **-help** Prints usage information on the standard error.

- **-picture** *picture_file* Names the file containing the picture to display.

- **-yes** *text* Label to confirm.

- **-no** *text* Label to cancel.

Exit Codes:

0 if confirm button is used.
1 if cancel button is used.

xdtinfo - file information

Syntax xdtinfo *file_list*

Description xdtinfo displays file characteristics and lets users modify them graphically.

Examples of X.Deskware Utility Usage

```
fyi -o "No static actions defined for that action"

gti -prompt "Enter filename:" -length 25 -step 5 while yni "continue?"; do date; done

xdtinfo /usr/*
```

For more examples, refer to the file */usr/lib/X11/xdt/xdtsysinfo*.

Application Defaults

The following X Window application default files come with the Desktop Manager:

Xhibit -application defaults: **xdt**

XDeskware -application defaults: **fyi, yni, xdtinfo, gti**

The deskware applications defaults have been designed to give the deskware utilities a Desktop Manager look and feel.

The application default files are automatically installed in the directory */usr/lib/X11/app-defaults*.

If you wish to override any of the application defaults for a specific user, this should be done by making entries in the *.Xdefaults* file in their home directory or use **xrdb** to load the properties into the server.

Sample Rule Files

The Desktop Manager includes the following sample rule files:

Desktop Manager System Rule File:

/usr/lib/X11/xdt/xdtsysinfo

Desktop Manager Directory Rule Files:

/bin/.xdtdirinfo

/etc/.xdtdirinfo

/usr/bin/.xdtdirinfo

/usr/bin/X11/.xdtdirinfo

Desktop Manager Initial Environment Files

/usr/lib/X11/xdt/xdtinitial.xde

Interprocess Links

The Desktop Manager uses the X InterClient Communication.

The **tellxdt** utility is included in this release for use in shell scripts or rule files. For example:

tellxdt [-l] [-a] [-n *name* [!*pid*]] *DCL_commands*

The **tellxdt** utility enables shell scripts or rules to send DCL commands to a running Desktop Manager. If there is more than one Desktop Manager process running, **tellxdt** sends the commands to the most recently executed Desktop Manager by default.

The options to **tellxdt** are as follows:

Table 7.21.

tellxdt Options

Option	Function
-l	Lists all active desktop processes.
-a	Sends commands to all active desktops.
-n	Sends to a named desktop process.

Appendix A
Setting Streams Parameters

This appendix explains how to display and set the streams parameters that apply to the X Window System.

Overview

A STREAM is a data transfer path between two or more processes. The X Window System uses the STREAMS MECHANISM to provide a communication path between clients and the server.

Streams use UNIX resources that are limited by values defined in KERNEL CONFIGURATION MODULES. Depending on the demand that you and other system users place on these resources, your system could run out of streams resources if you do not first reset the allocations in the kernel configuration modules.

Displaying Parameters

You can use the **crash** utility to display a listing of your system's current streams usage and parameters. This display tells you what the current streams settings are, as well as the extent to which each resource is being used. You can use this display to determine if your streams resources can be allocated more efficiently.

To use the **crash** utility from the Desktop window, execute the following steps:

1. Log in as root.
2. Open an Xterm window by double-clicking on the ODT-OS icon.
3. At the system prompt, enter **crash**.
4. At the utility prompt, enter **strstat**.

5. After looking at the resulting streams display, enter **q** to return to the operating system prompt.

The streams display called up by the **crash** utility contains the following headings:

Table A.1.

Streams Display Headings

Heading	Description
CONFIG	Number of streams currently configured.
ALLOC	Number of streams currently allocated.
FREE	Number of streams available for allocation. This number is the difference between CONFIG and ALLOC.
TOTAL	Number of attempted allocations since system start-up.
MAX	Highest number of streams allocated at one time since system start-up.
FAIL	Number of failures due to insufficient free streams since system start-up.

The default allocations that are established when you install Open Desktop are sufficient for most users. However, if you plan to run more than 10 programs simultaneously (and thus have at least 10 windows open at the same time), you may need to reallocate the streams resources. We recommend that you adjust these resources only if you are an experienced system administrator and understand the impact of these adjustments. Allocating too few streams resources could result in error messages, a locked-up server, or the inability to switch screens.

After you change any streams parameters, you must rebuild the kernel and reboot the system. These procedures are described in *Administering ODT-OS* in the *Administrator's Guide*.

Changing Parameters

Use the following procedure to reset streams parameters:

1. Log in as root.
2. Call up the **sysadmsh** program by double-clicking on the Sysadmsh icon.
3. At the prompt, enter the root password.
4. Select System from the SysAdmSh menu.
5. Select Configure from the System menu.
6. Select Kernel from the Configure menu.
7. Select Parameters from the Kernel menu.
8. Select category 11, "Streams Data," from the Parameters menu.

You can now specify any of the following parameters:

- Number of streams queues
- Total number of streams
- Number of streams buffers
- Number of streams pipes

Details about setting these parameters are provided in the following sections.

Number of Streams Queues

The total number of streams queues is defined by the parameter:

NQUEUE

The minimum recommended value is 192. You need 8 queues for each client-server connection. Each xterm client-server connection requires an additional 10 queues. For example, with 5 simultaneous clients, 2 of them xterms, you need (5 X 8) + (2 X 10) = 60 queues.

In general, when you run out of streams queues, a message is displayed on the console. When that happens, you should increase the NQUEUE value and rebuild the kernel.

Total Number of Streams

The total number of streams that may be opened at one time is defined by the parameter:

NSTREAM

The minimum recommended value is 48. Each client-server connection requires 2 streams. Each xterm connection requires an additional 2 streams. Using the example described previously (5 clients, 2 of them xterms), you need (5 X 2) + (2 X 2) = 14 streams.

As with streams queues, a message is displayed on the console when you run out of streams. When that happens, you should increase the NSTREAM value and rebuild the kernel.

Number of Streams Buffers

The default streams buffers quantities are listed in */etc/conf/cf.d/mtune*. Current values, if different from the default values, are found in */etc/conf/cf.d/stune*. As shown in the following table, the size of a buffer determines its recommended minimum quantity.

Table A.2.

Buffer Size Recommendations

Buffer	Minimum Number of Buffers
NBLK4096	4
NBLK2048	20
NBLK1024	20
NBLK512	8
NBLK256	16
NBLK128	32
NBLK64	128
NBLK16	40
NBLK4	40

In each buffer name, the number following "NBLK" is the buffer size. For example,

NBLK2048 20

means that you should have at least twenty 2048-byte buffers.

If you run out of buffers, the server freezes and cannot respond to any requests. If this situation arises, you must restart Open Desktop. To avoid this problem in the future, increase the values shown above, starting with a 50 percent increase in all values.

Number of Streams Pipes

The number of streams pipes is defined by the parameter:

NUMSP

The initial default is 32.

The NUMSP parameter affects the number of available streams, which in turn dictates the number of clients and xterms that you can run at the same time. Each client requires 2 streams pipes. Each xterm requires an additional 2 streams pipes. Using the example described previously (5 clients, 2 of them xterms), you need (5 X 2) + (2 X 2) = 14 streams pipes.

Rebuilding and Rebooting

After you have reset streams parameters, you must rebuild the kernel and reboot the system. For details about these procedures, see *Administering ODT-OS* in the *Administrator's Guide.*

Appendix B

Monochrome Configuration File

The following sample *.Xdefaults* file shows one way of configuring the X Window System if you have a monochrome monitor.

```
#
# SAMPLE .Xdefaults / app-defaults  RESOURCE SPECIFICATIONS FOR MWM
#
# MONOCHROME
#
#
# general appearance resources that apply to Mwm (all parts)
#

Mwm*font:                          hp8.8x16b

Mwm*backgroundTile:                background

Mwm*activeForeground:              Black
Mwm*activeBackground:              White
Mwm*activeTopShadowColor:          White
Mwm*activeTopShadowPixmap:         25_foreground
Mwm*topShadowPixmap:               25_foreground
Mwm*activeBottomShadowColor:       Black
Mwm*activeBottomShadowPixmap:      75_foreground
Mwm*bottomShadowPixmap:            75_foreground
Mwm*makeActiveColors:              false
Mwm*activeBackgroundPixmap:        50_foreground
Mwm*backgroundPixmap:              75_foreground
Mwm*menu*backgroundPixmap:         background

Mwm*foreground:                    Black
Mwm*background:                    White
Mwm*topShadowColor:                Black
Mwm*bottomShadowColor:             Black
Mwm*makeColors:                    false

Mwm*buttonBindings:                DefaultButtonBindings
Mwm*keyBindings:                   DefaultKeyBindings
```

```
Mwm*rootMenu:                          RootMenu
Mwm*windowMenu:                        DefaultWindowMenu

#
# general appearance resources that apply to specific parts of Mwm
#

Mwm*menu*topShadowPixmap:              25_foreground
Mwm*menu*background:                   White
Mwm*menu*topShadowColor                Black
Mwm*menu*bottomShadowColor             Black
Mwm*menu*makeColors:                   false

#
# Mwm - specific appearance and behavior resources
#

Mwm*positionOnScreen:                  false
                                       # prevents xterm downsizing on ega
Mwm*moveTreshold:                      40
Mwm*transientDecoration:               title
Mwm*useIconBox:                        true
Mwm*feedback*confirmbox*backgroundPixmap:  25_foreground

#
# General appearance and behavior defaults
#

*topShadowTile:                        foreground
*bottomShadowTile:                     foreground
*topShadowColor:                       White
*bottomShadowColor:                    Black
# *foreground:                         White
# *background:                         Black
*selectColor:                          White
*invertOnSelect:                       true
*borderWidth:                          5
*borderColor:                          Black

#
# END OF RESOURCE SPECIFICATIONS
#
```

Appendix C

Customizing Screen Colors

Screen display colors are assigned in the configuration file *$HOME/.Xdefaults*. You can customize the colors on your screen by editing *.Xdefaults* and changing the color entries of the relevant resources. When you have finished editing *.Xdefaults*, restart Open Desktop to enact the changes.

Any color name that is defined in the RGB database file */usr/lib/X11/rgb.txt* may be used. You can also modify the RGB database to define new colors or redfine existing ones. "Defining Colors in the RGB Database" in this chapter explains how to do this.

If you do not have an *.Xdefaults* file in your home directory, copy the system default configuration file */usr/lib/X11/app-defaults/Mwm* to *$HOME/.Xdefaults*. For more information on the *.Xdefaults* file, refer to "The .Xdefaults File," in this guide. The following sample *.Xdefaults* file shows one way of configuring the display with a color monitor.

```
#
# SAMPLE .Xdefaults / app-defaults  RESOURCE SPECIFICATIONS FOR MWM
#

#
# general appearance resources that apply to Mwm (all parts)
#
Mwm*font:                       fixed.snf

Mwm*backgroundTile:             background

Mwm*activeForeground:           Black
Mwm*activeBackground:           Cyan
Mwm*activeTopShadowColor:       LightCyan
Mwm*activeBottomShadowColor:    Black
Mwm*makeActiveColors:           false

Mwm*foreground:                 Black
Mwm*background:                 Gray
Mwm*topShadowColor:             White
Mwm*bottomShadowColor:          Black
```

```
Mwm*makeColors:                 false

Mwm*buttonBindings:             DefaultButtonBindings
Mwm*keyBindings:                DefaultKeyBindings
Mwm*rootMenu:                   RootMenu
Mwm*windowMenu:                 DefaultWindowMenu

Mwm*useIconBox:                 true
Mwm*showFeedback:               restart

#
# general appearance resources that apply to specific parts of Mwm
#

Mwm*menu*background:            Gray
Mwm*menu*topShadowColor:        White
Mwm*menu*bottomShadowColor:     Black
Mwm*menu*makeColors:            false

#
# Mwm - specific appearance and behavior resources
#

Mwm*positionOnScreen:           false     # prevents xterm downsizing on ega
Mwm*moveTreshold:               40
Mwm*transientDecoration:        title   # no resize frame for popup windows

#
# END OF RESOURCE SPECIFICATIONS
#
```

Defining Colors in the RGB Database

The colors available on your system are defined in the RGB database file */usr/lib/X11/rgb.txt*. Each line of *rgb.txt* consists of three color values and a color name. The color values are decimal numbers from 0 to 255 for the red, green, and blue components of the color. A sample line from the *rgb.txt* file looks like this:

```
      35        35        142         Navy Blue
```

This entry defines navy blue as consisting of 35/255ths of the maximum possible intensity of red, 35/255ths of the maximum possible intensity of green, and 142/255ths of the maximum possible intensity of blue.

To define (or redefine) colors in the RGB database:

- Log in as **root**
- Edit */usr/lib/X11/rgb.txt*
- Recompile the RGB database by entering the command

  rgb rgb < rgb.txt

Appendix D
Changing Video Systems

This appendix explains the steps that you must take when changing monitors and video adapter cards.

Overview

Whenever you change video systems, you must edit one or more configuration files to ensure that Open Desktop's windows and icons appear correctly on your screen. The number and types of files that must be edited depend on what type of video system you currently have and what type you plan to use. To help you edit the configuration files, Open Desktop comes with two CONFIGURATION SCRIPTS that automatically edit the appropriate files based on information that you provide about your new monitor and adapter card.

Description of the Configuration Scripts

The configuration scripts provided with Open Desktop are named *mkdev graphics* and *xconfigure*. The following sections describe what each script does and when each should be used.

mkdev graphics

The *mkdev graphics* script lets you choose and activate one of several *graphinfo* files. Each *graphinfo* file contains descriptive information about a particular video adapter card. This information is used by the X Window System server whenever you start up Open Desktop. You should run *mkdev graphics* whenever you:

- Install a different video adapter card
- Change the mode of your current video adapter card

Examples of these situations are given later in this appendix.

xconfigure

The *xconfigure* script changes the settings in the X Window System's */usr/lib/X11/app-defaults* files. These settings control the appearance of Open Desktop's windows and icons, and must be changed whenever you change monitors. You should run *xconfigure* whenever:

- You install a different video adapter card
- You install a different monitor
- Windows or icons do not appear correctly on your screen.

Examples of these situations are given later in this appendix.

Running the Configuration Scripts

The following sections describe how to run *mkdev graphics* and *xconfigure*.

mkdev graphics

To run *mkdev graphics*, execute the following steps:

1. Log in as root.
2. At the prompt, enter **mkdev graphics**.
3. Select Form from the graphics menu.
4. Select your adapter type from the point and pick list and enter it in the Adaptor type field of the Video Configuration form.
5. Select your adapter mode from the list and enter it in the Adaptor mode field of the Video Configuration Form.
6. Select Accept on the Video Configuration Form to reconfigure the appropriate files based on the choices you just made,

   -or-

   select Ignore to start over at the beginning of the Video Configuration Form.

If your video card's type and mode are not on the point and pick lists, refer to the documentation that came with the card. In most cases, the card's manufacturer provides the specifications that must be included in the *graphinfo* file for their particular card. In this situation, you must manually edit *graphinfo* so that it contains the data provided by the card manufacturer.

If your video card's type and mode are not on the point and pick lists, but your card is fully register-compatible with an IBM® video card, choose the IBM selections from the point and pick lists.

xconfigure

To run *xconfigure*, execute the following steps:

1. Log in as root.
2. At the prompt, enter **/usr/bin/X11/xconfigure.**
3. Select either Color or Monochrome.

In addition to differentiating between color and monochrome monitors, this script also automatically distinguishes between VGA and EGA color monitors.

When you run *xconfigure*, any changes that you made previously in *.Xdefaults* are overwritten. If you want to save any of these changes, you must move them to another file before running the configuration script, and then manually put them back into *.Xdefaults* after running the script.

Examples

The following examples show which scripts to use when changing various combinations of video system components. The following convention is used to describe card and monitor resolution:

640 x 480 x 16

The first number is the horizontal screen measurement (in pixels); the second number is the vertical screen measurement (in pixels); the third number is the maximum number of colors that can be simultaneously displayed. The example above refers to a video system that can display images measuring 640 x 480 pixels, and that up to 16 colors can be displayed at the same time.

Changing Monitors

Suppose that your system currently uses a VGA monitor with a resolution of 640 x 480 x 16, and that your video adapter card supports extended resolutions of up to 800 x 600 x 256. You can take advantage of the extended resolution supported by the card by changing just your system's monitor. In this situation, you should run the *mkdev graphics* script (because you are changing the adapter's mode) and the *xconfigure* script (because the */usr/lib/X11/app-defaults* files must be reconfigured due to the monitor change).

Changing Video Adaptor Cards

Suppose that your system currently uses a monitor with a resolution of 800 x 600 x 256, but that its video adapter card only supports resolutions of 640 x 480 x 16 or less. If you install a new adapter card to take advantage of the monitor's resolution, you must run the *mkdev graphics* script (because you are changing adapter cards) and the *xconfigure* script (because the */usr/lib/X11/app-defaults* files must be reconfigured due to the card change).

Changing Monitors and Video Adaptor Cards

Suppose that your system currently uses a monochrome monitor and adapter card. To upgrade to a color video system, you must run the *mkdev graphics* script (because you are changing adapter cards) and the *xconfigure* script (because the */usr/lib/X11/app-defaults* files must be reconfigured due to the monitor change).

Administering ODT-OS

12/21/89-1.0.0D
Processed: Wed Dec 20 10:56:24 PST 1989

Contents

Chapter 1
Introduction

The ODT-OS operating system is SCO UNIX System V/386. Your UNIX system is a collection of programs that allows you to accomplish a full spectrum of tasks, from developing high-level and assembly language programs to creating, editing, and typesetting documents. To keep running smoothly, the system requires careful control of its operation and a regular schedule of maintenance. This guide explains how to operate and maintain the operating system on your computer, ensuring maximum performance with the fewest system problems.

An important part of system operation is the protection of data on the system. Security is discussed in great detail in this guide; the operating system includes flexible mechanisms designed to protect your data.

The System Administrator and Administrative Roles

Every UNIX system should have at least one person in charge of system maintenance and operation. In this guide, such persons are called system administrators. It is the task of system administrators to ensure the smooth operation of the system and to perform tasks that require special privileges. These duties require that the system administrator(s) become proficient with a wide variety of functions.

Depending on the size of the system and the number of users on it, system administration can be anything from a once-a-day task to a full-time job. Even if the system is small, the system administrator should faithfully perform each required maintenance task, since sloppy maintenance can adversely affect system performance.

The system administrator should keep a log of all system modifications and system events. Each event, message, backup, or modification should be logged with the date, time, and name of the person logging, and the circumstances surrounding the event. For example, if a new application is added to the system software, an entry should be placed in the log. This entry should include the time, date, and name of the person installing, and any notes about the software or installation that may be helpful. An accurate log helps in diagnosing system problems and charting the growth and use of a system.

All tasks in this guide are presented from a system administrator's point of view, but many can also be accomplished by ordinary users. Since some of the tasks dramatically change the system's operation, we recommend that, whenever possible, the system administrator perform these tasks. However, no matter who performs an operation, it should be logged in the system log. Following these rules can prevent unwanted or unnecessary changes to the system.

A system administrator has several tasks to perform, sometimes on a daily basis:

- Making certain that adequate backups (regular copies of files on the system) are made and stored for future use.
- Handling problems related to use of limited computer resources (disk space, number of processes, etc.)
- Alleviating system communication (network) stoppages due to failed connections.
- Applying operating system updates and maintenance fixes.

Making Administration Easier with the sysadmsh

The **sysadmsh** is a menu interface designed to simplify the task of system administration. The menus, submenus and screens allow you to simply point and pick, or fill in blanks. The **sysadmsh** allows less-experienced system administrators to use UNIX commands that would otherwise require memorization and constant referring to manual pages. The **sysadmsh** includes context-sensitive help; simply press the **F1** key from any menu to display further explanations of the menu options.

If you are new to UNIX operating systems, we strongly recommend that you become familiar with the concepts and tasks covered in *Using ODT-OS* in the *User's Guide*, a tutorial for new users. This guide assumes some familiarity with UNIX systems; after studying *Using ODT-OS*, you should be able to perform the basic system administrative tasks described here.

To aid users of **sysadmsh**, the documentation of this guide is supplemented by **sysadmsh** references that appear below UNIX command line instructions.

For example, the following instructions refer to the **custom** utility, used to add more software to your system. Below the actual command is a sequence of **sysadmsh** menu selections.

Enter the following command:

custom

Δ **sysadmsh** users select: System→Software

This means that you can access the functions of the **custom** command by first returning to the desktop and double-clicking on the Sysadmsh icon, followed by selecting "System" at the main **sysadmsh** menu, followed by selecting "Software" at the next lower level. Selections can be made from the menu in any of the following ways:

- Tab through the menu options using the **Space** key and press **Return** on the option you want.
- Move left and right through the options using the arrow keys and press **Return** on the desired option.
- Press the first letter of the option desired. This is the quickest way. Using the example above, you would simply enter "ss" (without the **Return** key) to reach the **custom** menu.

For more instructions on using the **sysadmsh**, refer to the "sysadmsh: Using the System Administration Shell" chapter in this guide.

The Super User Account

The super user account is a special account for performing system maintenance tasks. It gives the system administrator unusual privileges that ordinary users do not have, such as accessing all files in the system, and executing privileged commands. Many of the tasks presented in this guide require that the system administrator be logged in as the super user. To do this, the system administrator must know the super user password created during the installation of your system. (See the *Installation Guide*.)

Log in as the super user only to perform system-maintenance tasks. Even if the system administrator is the only one using the system, that person should create a user account for day-to-day work, reserving the super user account for system-maintenance tasks only.

Few users should know the super user password. Misuse of the super user powers by naive users can result in a loss of data, programs, and even the operating system itself.

The Keyboard

Many keys and key combinations have special meanings in UNIX operating systems. These have names that are unique to UNIX systems, and may not correspond to the keytop labels on your keyboard. To help you find these keys, the following table shows which keys on a typical terminal correspond to those on UNIX systems. A list for your particular login device is in **keyboard**(HW).

In this table, a hyphen (-) between keys means "hold down the first key while pressing the second."

Table 1.1.

Special Keys

UNIX Name	Keytop	Action
	Delete	Stops the current program, returning to the shell prompt. This key is also known as the INTERRUPT or **Del** key.
Backspace	Backspace	Deletes the character to the left of the cursor.
Ctrl-d	Ctrl-d	Signals the end of input from the keyboard; exits current shell or initiates the "logout" procedure if the current shell is the login shell.
Ctrl-h	Erase	Deletes the first character to the left of the cursor. Also called the ERASE key.
Ctrl-q	Ctrl-q	Restarts printing after it has been stopped with Ctrl-s.

(Continued on next page.)

Table 1.1.

Special Keys *(Continued)*

UNIX Name	Keytop	Action
Ctrl-s	Ctrl-s	Stops printing at the standard output device, such as a terminal. Does not stop the program.
Ctrl-u	Ctrl-u	Deletes all characters on the current line. Also called the KILL key.
Ctrl-	Ctrl-\	Quits current command and creates a *core* file. (Recommended for de-bugging only.) See **core**(F) for more information.
Esc	Esc	Exits the current mode; for example, exits insert mode when in the editor **vi**.
Return	Return	Terminates a command line and initiates an action from the shell.

Many of these special function keys can be modified by the user. See **stty**(C) for more information.

About This Guide

The tasks presented in this guide range from simple ones requiring very little knowledge about UNIX systems, to complex tasks requiring extensive knowledge about the operating system and your computer.

Each chapter explains the tools and knowledge you need to complete the tasks described in that chapter. In some cases, you may be referred to other manuals.

This guide contains chapters about computer hardware you may wish to use with your system. The use and interaction of various devices with the operating system is described in a comprehensive fashion. For example, "Using Floppy Disks and Tape Drives" discusses the use of magnetic storage media, and covers the basics of preparing the operating system for such a device, installing it, and how to use the drive once it has been installed.

In addition, there are chapters dealing with several other types of devices you may wish to use, and there are many chapters to help you administer your system. Some are designed to help you set up a network with other computer systems and some help you maintain and understand your own system.

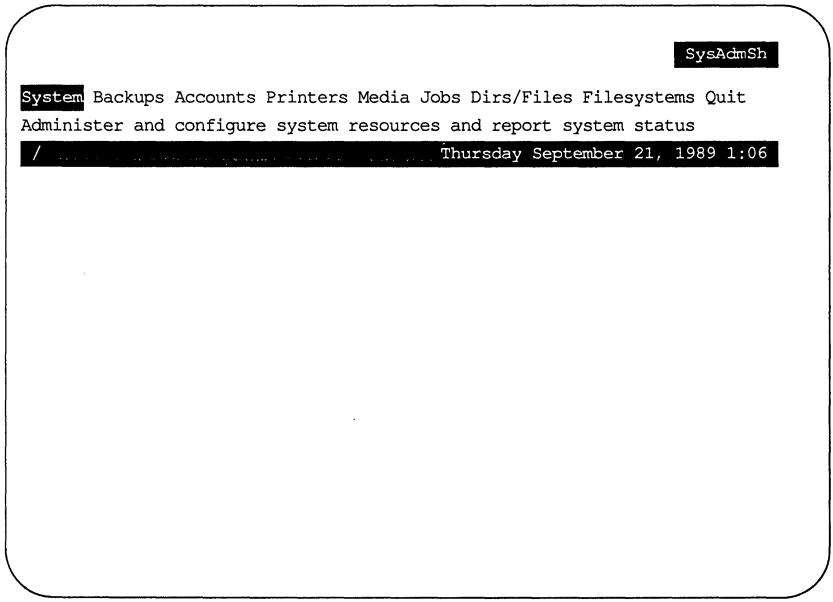
Chapter 2

Using the System Administration Shell

The **sysadmsh** (**sys**tem **adm**inistration **sh**ell) is a menu interface designed to simplify the task of system administration. You will find it easier to learn the material in this chapter if you start the **sysadmsh** and actually run the examples as you get to them. You should become familiar with the concepts covered in the *Using ODT-OS* in the *User's Guide* before using the **sysadmsh** menus.

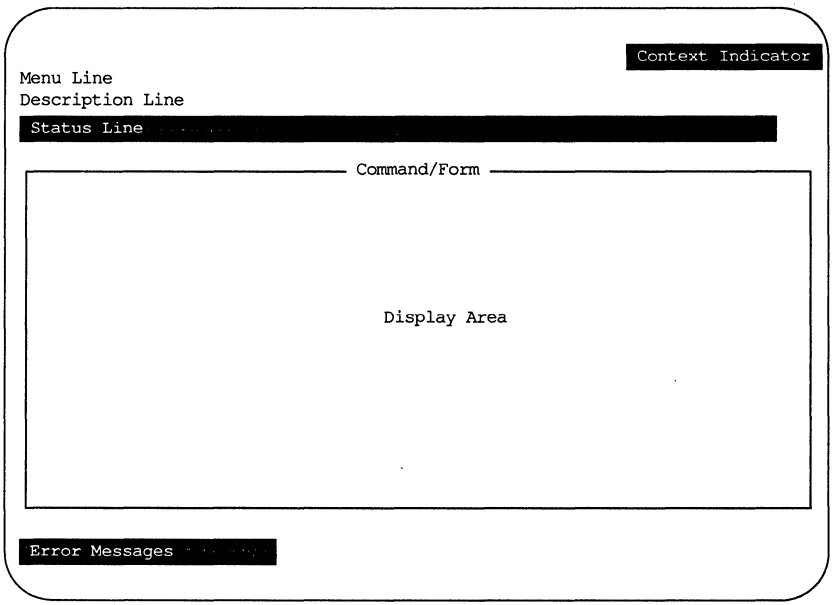
Starting sysadmsh

For the purposes of this tutorial, double-click on the Sysadmsh icon.
The main **sysadmsh** menu is displayed:



How the Screen is Organized

A schematic of the **sysadmsh** screen is given below. Areas shown in black appear on the screen as highlighted areas or bars of text. Each area is used to display specific types of information:



- The Context Indicator is the highlighted bar of text in the upper-right corner of your screen. It displays the name of the current menu. The Context Indicator for the **sysadmsh** opening screen shows *SysAdmSh*.
- The Menu Line displays the menu options that are currently available. The main **sysadmsh** menu consists of nine options: System, Backups, Accounts, Printers, Media, Jobs, Dirs/Files, Filesystems, and Quit.
- The Description Line gives you a brief description of the currently highlighted menu option.
- The Status Line is the highlighted bar of text that separates the Menu and Description Lines from the Display Window. The Status Line in the **sysadmsh** opening screen contains the date, time, and current working directory. When a UNIX command is being executed, the name of the command and the options being used are displayed at the far left of the Status Line.

- The Command/Form line displays a title for the contents of the Display Area. This can be either a UNIX command name, or the name of a **sysadmsh** form.
- The Display Area is where **sysadmsh** forms and Scan Windows are displayed. Forms and Scan Windows are explained in detail later in this chapter.
- Error Messages and recovery instructions are displayed on the last line of the screen in highlighted text.

Selecting Menu Items

The keystrokes listed in Table 2.1 are used to traverse the menus. Note that there are several ways to select options; if you have used menu-based programs before, use the method you are most familiar with.

Table 2.1.

Basic Menu keystrokes

Action	Keystroke
Move to menu option	Arrow keys, or **Space** (same as right arrow)
Select menu option	First letter of option, or move highlight to option and press **Return**
Retreat to previous menu	**Esc**
Get help	**F1**

You can familiarize yourself with the menu options by using the Arrow keys or **Space** to move the highlight from option to option. Each time you move the highlight to a new option, a description of that option appears on the Description Line.

sysadmsh has a hierarchical menu structure. Many of the menu options move you down to another menu. For example, when you select the Processes option from the main menu, a sub-menu containing more options is displayed which lets you check on and manipulate your machine's processes. The menu hierarchy makes it easy to find the command you need by moving down from one menu to the next. Eventually you get to a menu option that either executes a UNIX command or displays a form that you must fill in with the details that the command needs. Note that typing the first letter of the option name is the quickest way to move through menu levels; in time you will be able to instantly reach the function you need by pressing three- and four-letter codes you have memorized.

The best way to learn how to use menus is to practice making menu selections with these keystrokes. If you select an option by mistake, you can always retreat to the previous menu by pressing the **Esc** key. If you are several levels deep, you can return to the Main menu by pressing the **F2** key and then typing **n**. **F2** takes you to the Quit option, and **n** returns you to the Main menu. To help you find your way through the **sysadmsh** menus, Table 2.2 contains a map of the second level menus.

Table 2.2.

Menu Map

System	Backups	Accounts	Printers	Media	Jobs	Dirs/Files	Filesystems	Quit
↓	↓	↓	↓	↓	↓	↓	↓	↓
Report	Create	User	Configure	List	Report	List	Check	Yes
Configure	Restore	Defaults	Schedule	Extract	Terminate	View	Mount	No
Hardware	Schedule	Terminal	Request	Archive	Authorize	Copy	Umount	
Software	View	Report	Auxiliary	Format		Edit	Add	
Audit	Integrity		Priorities	Duplicate		Modify	Floppy	
Execute				Tapedump		Print	DOS	
Terminate						Archive		
						Differences		
						Remove		
						UseDOS		

This chapter uses a syntax convention for denoting a string of menu options. For example, to print a file you must select the Dirs/Files option from the main menu, and then select the Print option from the Dirs/Files menu. This sequence is denoted by the shorthand notation Dirs/Files→Print, and can be executed by typing **dp**.

When you select a menu option, one of three things happens:

- A lower lever menu is displayed,
- You are dropped into a form, or
- A UNIX command is executed and the result displayed in a Scan Window.

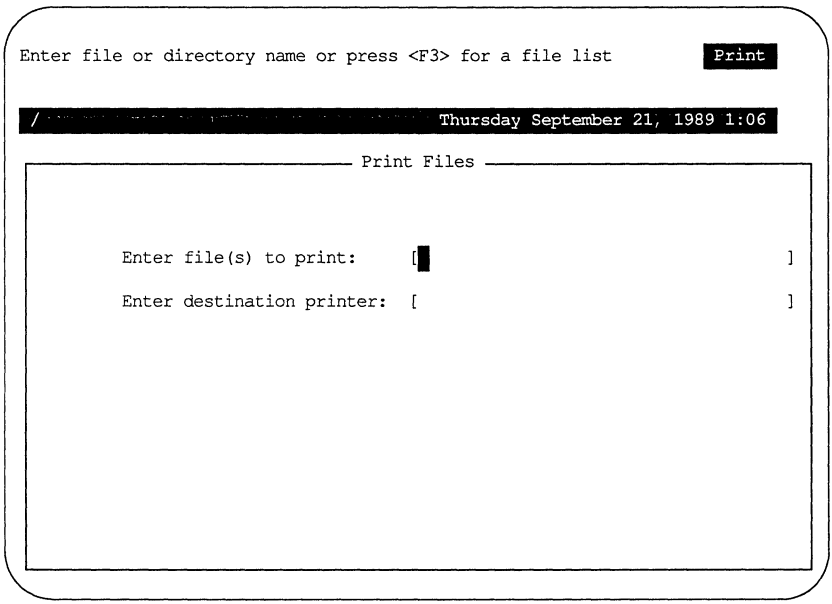
The next two sections explain the details of Forms and Scan Windows.

Using Forms

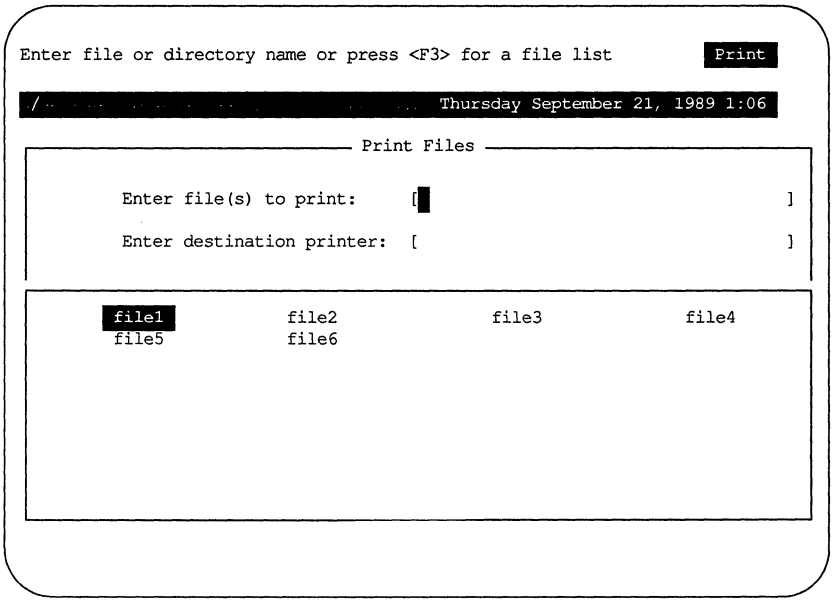
Some menu options require additional information in order to perform the correct task. For example, the **Print** option cannot do anything until you tell it what you want to print and which printer to use. When you select an option of this sort, a form appears on the screen. By filling in the form, you give the command the information it needs.

The example below demonstrates how forms work by showing you how to print a file in your current directory. After the example, the keystrokes are listed that allow you to move around the form, edit it, and select "Point and Pick" choices.

To print a file, first select Dirs/Files→Print. The Print form is displayed:



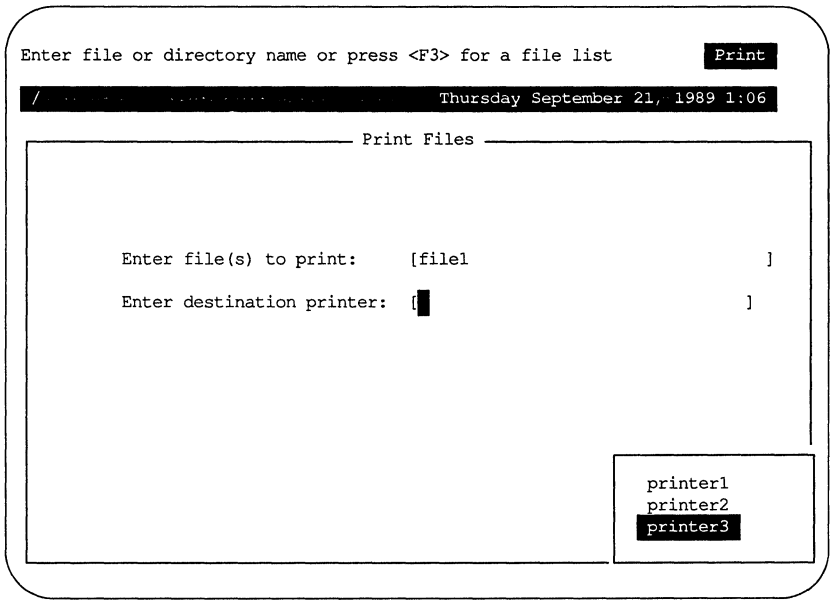
Notice that the highlight is on the first item in the form. You can fill in the field or obtain a list of choices by pressing **F3**. You can enter the filename if you know it, but for the sake of this tutorial, assume you need to find the filename and press **F3** now. A window opens up overlapping part of the Print form:



The window contains a list of files that you can select. To select a file, "point" to it by moving the highlight to it, and "pick" it by pressing return. This is known as "Point and Pick," and is used whenever a range of choices is displayed. When you have made your selection, the window closes and you are returned to the Print form.

Note that the name of the file you selected is now displayed in the form. You can now change the name using the edit keys (listed later in this section), or press **Return** to move to the next field.

Now you should enter the name of the printer to be used. If you don't know the printer name, press **F3** and another, smaller window is opened that contains a list of installed printers:



You can select the printer just as you did the name of the file. When you have selected a printer, you are returned to the Print menu.

The keystrokes listed in the following tables allow you to use forms easily.

Table 2.3.

Form Keys

Keystroke	Action
Esc	Tells the program that you have changed your mind and do not want to finish filling in this form. The form is removed, and no action performed. You are returned to the previous menu. In addition, **Esc** followed by **Return** is used to acknowledge that an error message has been read and that you are ready to continue.
Up, Down Arrow	Moves to other fields in a form. Some fields are restricted and no input is allowed. The Arrow keys will skip over these. Other fields must be filled in. Pressing the Down Arrow key on the last item in a form brings you back to the first item.
Left, Right Arrow	Moves left and right in the current field. Allows changing of text without retyping the entire line.
Return	Pressing **Return** on a field completes the data entry to that field, and moves the cursor to the next field. In the last field, **Return** completes the entire form and tells the shell that the data is ready to use.
Ctrl-x	Exits and executes the form from wherever you are. Think *x* for *execute*. **F10** does the same.
F4	You can use the spelling checker utility when you are in a form. If you think a word might be misspelled, press **F4** while the cursor is on the word and a list of possible correct spellings will appear in a Point and Pick list. The word you select replaces the misspelled word.

Table 2.4.
Edit Keys

Keystroke	Action
Ctrl-y	Delete the current line, start over.
Ctrl-w	Delete the current word.
Ctrl-g-Ctrl- h	Move the cursor to the beginning of the line.
Ctrl-g-Ctrl- l	Move the cursor to the end of the line.
Ctrl-v	Toggle into or out of overstrike mode.
Del	Deletes character over the cursor.
Backspace	Back up and delete one character (left of cursor).
Ctrl-u	Page up.
Ctrl-d	Page down.
Ctrl-n	Next word.
Ctrl-p	Previous word.
Left, Right Arrow	Move left and right within the edit line.

Table 2.5.

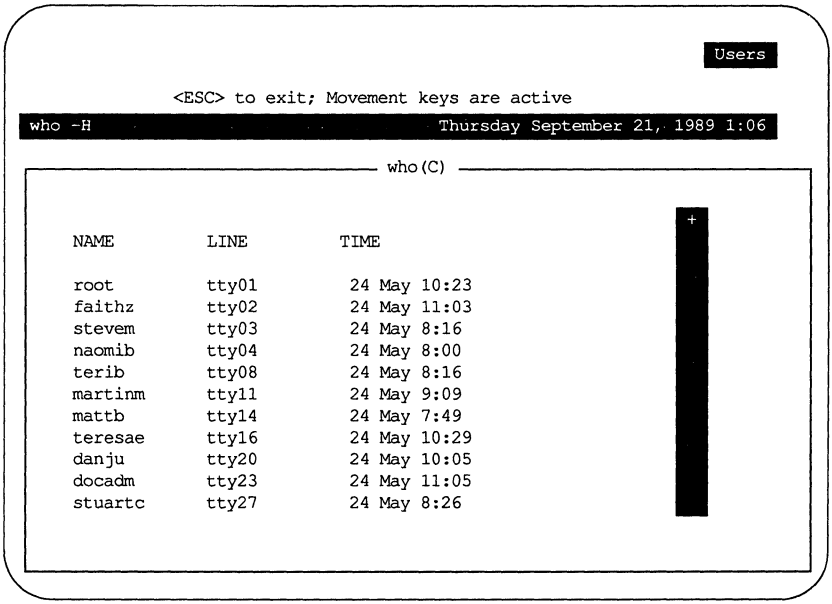
Point and Pick Keys

Keystroke	Action
Return	Pressing **Return** on a item name selects the item.
Esc	No item is desired, abort the selection process. The list is removed and no action performed.
Ctrl-v	Toggles between selecting all or none of the items appearing in a list.
Up, Down Arrow	Move to other items in a list.
Left, Right Arrow	Move across a multicolumn display.
Space	When the application will accept more than one item, the space bar is used to mark them. A marked item is indicated by a "*" character in the left column. It may be unmarked by pressing the space bar a second time while on the item. The entire collection of marked items is selected by pressing **Return**.
F5	The "Search" key is useful for finding items in long listings. A prompt appears and you enter the string to search for, and press **Return**. If the item is found, the highlight moves to that item, and another **Return** selects the item. If no match is found the highlight does not move. The ; and : keys repeat the previous search forward and backward, respectively.
First letter	The fastest method of selecting an item is by its first letter. Press the first letter of the item and the highlight moves to that item. Pressing **Return** then selects the item. (If there is only one item beginning with that letter, it will be marked by typing its first letter. There is no need to hit **Return** again.) If several items begin with the same letter, the cursor will move to the first occurrence in the list.

Using Scan Windows

When you execute a UNIX command by selecting a **sysadmsh** menu option, the result of the command is typically displayed in a Scan Window. Scan Windows are also used to display the contents of files and directory listings. To demonstrate the use of Scan Windows, let's say you want to know who is currently logged on to the system. To do this, select System→Report→Users. (This runs the UNIX **who**(C) command.)

When you select the Current option, a Scan Window that displays the output of the **who**(C) command appears in the Display Area:



Note that the name of the command (**who**) and the reference section in which its description can be found (C) are displayed at the top of the window. Also note that the option given to the command (-H) is displayed in the right hand side of the Status Line. If you do not understand the information displayed, look up the proper manual page for more information.

You can recognize a Scan Window by the vertical "scroll bar" that appears at the extreme right edge of the window. When the window is at the top of your text, the plus symbol (+) at the top of the Scroll Bar is visible. If it is at the bottom, the plus symbol at the bottom of the Scroll Bar is visible. You can see both plus signs when the window contains all of the text.

The scroll bar also indicates where you are in the window. The highlighted portion of the bar represents the section of text that is currently displayed in the window. As you scroll up and down, the highlighted bar moves with you.

Use the keys listed in Table 2.6 when you are in a Scan Window.

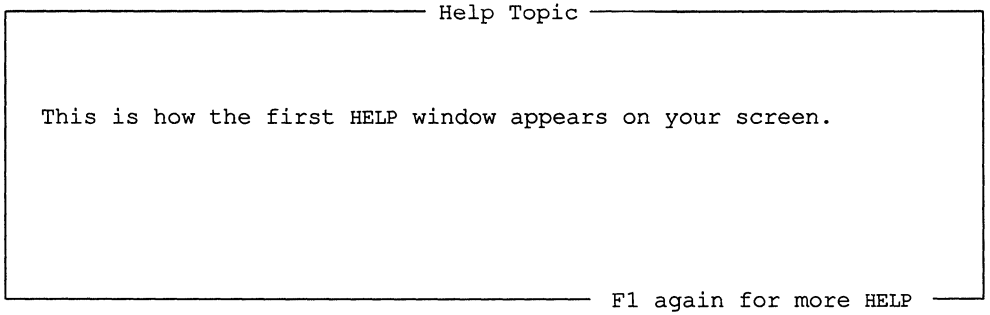
Table 2.6.

Scan Keys

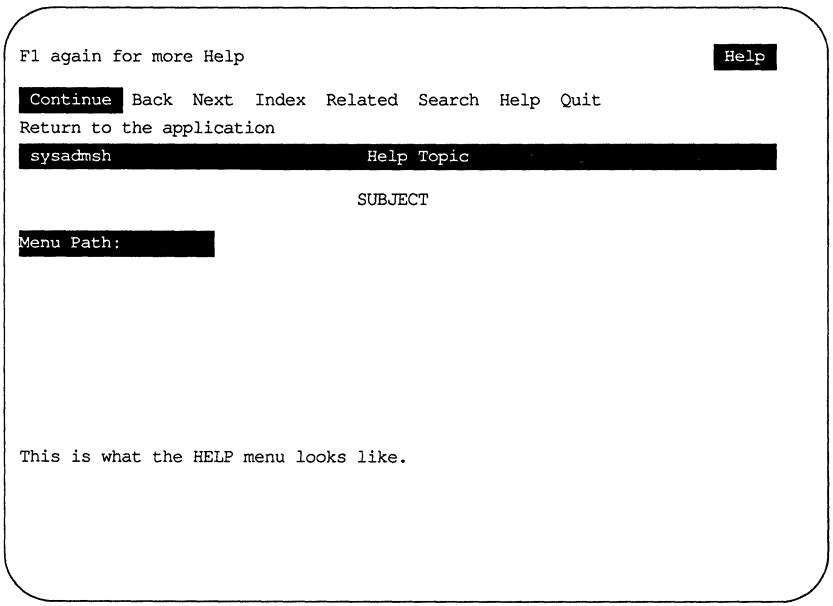
Action	Keystroke
Exit the file	**Esc**
Move up one line	Up Arrow
Move down one line	Down Arrow, or **Return**
Move down a page	**PgDn**, or **Space**
Move up a page	**PgUp**
Move to top of display	**Home**
Move to bottom of display	**End**
Search for a pattern in display (; and : repeat the search forward and backward, respectively)	**F5**
Print the output of command or file currently in Scan Window	**F7**

Getting Help

You can press the **F1** key to display more information to help you with your selection. When you press the **F1** key, a Help window opens within your current screen. It looks like this:



The window contains some basic information. If you need more help, you can press **F1** again and the complete Help menu is displayed:



From this menu you can select a variety of more detailed information. When you are finished, select Quit from the Help menu and you will be returned to your place in the **sysadmsh** menu proper.

The menu options for Help are listed in Table 2.7.

Table 2.7.

Help Options

Option	
Continue	Continue on to the next page. All the vertical movement keys are active: Up and Down Arrows, Page Up and Down, Home and End. If there is no further information, the highlight moves to the Help menu Quit option and the Description Line reads "Return to the application."
Back	Move back to topics that have been seen previously. There is no corresponding "Forward." This is also used to back up to more general topics. You can go back until the top-level introductory topic is reached.
Index	Choose a new topic from a list of indexed topics.
Related	Choose a new topic related to the current one.
Search	Search for a new topic by matching a pattern. First, you specify where to look (the titles, the text lines or both), and then give the pattern. The pattern can be a simple keyword (like "create" or "date") or a more complex "regular expression." A list of topics containing the pattern is presented.
Help	How is the help facility itself used? A table similar to this one is displayed on the screen. If you need further information, look for your topic in Index, Related, or Search.
Quit	Exit Help and return to **sysadmsh**. **F2** or **Esc** are other ways to exit quickly.

Each Help screen has general information available, as well as specific information about each option listed on the menu from which Help was selected. Each descriptive passage is preceded by the associated Menu Line and followed by a reference to the operating system documentation.

NOTE: When you are within an actual UNIX command, you do not have access to the Help facility. For example, when you select: Dirs/Files→Edit, you are within the UNIX **vi** command, and the **sysadmsh** keys no longer function. When you exit the command and return to the **sysadmsh**, the keys will function as expected. If no element of the **sysadmsh** is visible on the screen (Menu Line, boxes, Context Indicator, etc.) then Help will probably not be available. If you need help, exit from the current process and press the **F1** key to view Help. In general, it is best to use Help prior to executing a menu selection.

The Function Keys

The function keys give you access to several time-saving features.

Table 2.8.

Function Keys

Key	Action
F1	Help key - displays help for the current context within the application. Further information is available by pressing **F1** again.
F2	Exit key - activates the Quit option on the top menu-level. Press 'n' to return to **sysadmsh**.
F3	Pop-up key (used within a form) - displays a list of items that are acceptable for the current field.
F4	Spell key (used within a form) - displays a list of words that are possible correct spellings of the word in the current field. Select a word from the list by pressing **Return**. The word is then placed in the field.
F5	Search key (used within a window) - prompts for a string to search for. When you enter a string and press **Return**, the highlight moves to the item in the list that matches the pattern. If no match is found, the search fails and the highlight does not move. In addition, the semicolon (;) is used to repeat a search forward and the colon (:) to search backward.
F6	New directory key - offers the opportunity to change your current working directory. Note that this will not change the directory you were in when you invoked **sysadmsh** after you leave.
F7	Print key - prints the output of any command which is displayed in a Scan Window.

Chapter 3

Starting and Stopping the System

This chapter explains how to start and stop your system. It also explains how to log in as the super user (root), how to change the system startup/boot procedure, and use information displayed at boot time.

Starting the System

Starting a UNIX system requires more than just turning on the power. You must also perform a series of steps to initialize the system for operation. Starting the system requires:

- loading the operating system,
- checking the filesystems (if the system was improperly stopped), and
- choosing the mode of system operation.

The following sections describe each of these procedures.

Loading the Operating System

The first step in starting the system is to load the operating system from the computer's hard disk. Follow these steps:

1. Turn on power to the computer and hard disk. The computer loads the UNIX bootstrap program and displays the message:

```
SCO System V/386

Boot
:
```

2. Press the **Return** key. The bootstrap program loads the operating system.

When the system is loaded, it displays information about itself and checks to see if the "root filesystem" (that is, all files and directories) is in order and not corrupted. If a filesystem is uncorrupted and in good order, it is called "clean." If the root filesystem is clean, you can choose the mode of operation. If not, the system requires you to clean the filesystem before choosing.

Cleaning the Filesystem

You must clean the filesystem if the system displays the message:

```
fsstat: root filesystem needs checking
OK to check the root filesystem (/dev/root) (y/n)?
```

This message is displayed only if the system was not stopped properly, as described in the section "Stopping the System" later in this chapter.

This message is generated for each filesystem. The operating system requires clean filesystems to work properly. If the above message does not appear, your filesystem is clean and ready to use.

To clean the filesystem, enter **y** (for "yes") and press the **Return** key. The **fsck**(ADM) utility cleans the filesystem, repairing damaged files or deleting files that cannot be repaired. It reports on its progress as each step is completed. At some point, you may be asked if you wish to salvage a file. Always answer by entering **y** or **n** and pressing the **Return** key. For an explanation of how **fsck** works, see the "Filesystem Integrity" section of the "Using Filesystems" chapter in this Guide.

When cleaning is complete, the system asks you to choose the mode of operation.

Choosing the Mode of System Operation

You may choose the mode of operation as soon as you see the message:

```
INIT: SINGLE USER MODE

Type CONTROL-d to continue with normal startup,
(or give the root password for system maintenance):
```

The system has two modes: *normal operation* and *system maintenance*. (Normal operation is known as *multi-user mode*, while system maintenance mode is known as *single-user mode*.) Normal operation is for ordinary work on the system. This is the mode that allows multiple users to log in and begin work. System maintenance mode is reserved for work to be done by the system administrator. It does not allow multiple users.

To choose normal operation, press **Ctrl-d**. The system displays a startup message, you are prompted to enter the system time (see the next section) and the system executes commands found in the */etc/rc** scripts, generating startup messages for the various system services, such as the printer or network services. (These scripts are described later in this chapter.) Next, the system displays the "login:" prompt. You may then log in as a normal user, or as the super user, as described later.

To choose system maintenance mode, enter the super user password (also called the "root password") and press **Return**. You are then asked to set the system time, followed by the super user prompt (#). The commands in the */etc/rc** scripts are not executed. (Choose system maintenance mode only if you must do system maintenance work that requires all other users to be off the system.) When you log out of system maintenance mode using **Ctrl-d**, the system automatically enters normal operation.

Entering System Maintenance Mode by Shutting Down

To go from normal operation to system maintenance mode, log in as **root** and give the following command to shut down the system:

/etc/shutdown -g*n*

Δ **sysadmsh** users select: System→Terminate

Where *n* is the number of minutes until multiuser mode is stopped.

NOTE: If you attempt to select **System→Terminate** from within an X-window, (as opposed to executing **sysadmsh** from the UNIX command line) it will fail. The easiest way to shut down the system is to log in as **root** and use the **shutdown** command.

Entering System Maintenance Mode Directly

To go from normal operation to system maintenance mode directly, log in as **root** and give the following command:

/etc/shutdown -g2 su

The **su** indicates that you want to go directly into single-user mode rather than shut the system down.

NOTE: There is no **sysadmsh** equivalent for this command.

Setting the Time and Date

Once normal operation starts, the system asks for the correct time and date. It displays what it believes is the current time and date and then the following message:

```
INIT: New run level: 2

Current System Time is Wed Nov 29 08:19:00 PST 1989
Enter new time ([yymmdd]hhmm):
```

Unless your clock battery has been drained or removed, there should be no need to change the date. To leave the time and date unchanged, simply press **Return**. If you need to change the time and date, enter the new time and press **Return**. The new values must be entered as two or more consecutive pairs of digits, where the digits may be one or more of the following:

yy	(Optional) Represents the current year. It may be any two-digit value, from 00 to 99 for the years 1900 to 1999, respectively.
mm	(Optional) Represents the current month. It may be any two-digit value, from 01 to 12 for the months January to December, respectively.
dd	(Optional) Represents the current day. It may be any two-digit value, from 01 to the last day of the month.
hh	Represents the current hour. It may be any two-digit value, from 00 to 23. Hours are expressed in military time, where morning hours range from 00 to 11 and evening hours from 12 to 23.
mm	Represents the current minutes. It may be any two-digit value, from 00 to 59.

For example, to change the time and date to February 3, 1901 at noon, enter:

0102031200

and press **Return**. values, the system displays the new time and date:

Tue Jan 1 12:00:01 PDT 1901

If you enter an incorrect value, the system prompts you to try again. If you do not enter an optional value, the current value for that item remains unchanged. If you type a new value for the year, you must also type values for the month and day. Similarly, if you type a new value for the month, you must type a value for the day.

The time and date is followed by service startup messages and the "login:" message.

Logging in as the Super User

Many system maintenance tasks, when performed during normal operation, require you to log in as the super user. For example, you must be logged in as the super user to stop the system.

To log in as the super user, you must know the super user password. You also need to see the "login:" message on your terminal's screen. If you do not see this message, press **Ctrl-d** until it appears.

To log in as the super user, follow these steps:

1. When you see the "login:" message, enter the super user's login name:

   root

   Now press the **Return** key. The system prompts you for the super user's password.

2. Enter the super user's password and press the **Return** key. The system does not display the password as you enter it, so enter each keystroke carefully.

The system opens the super user account and displays the message of the day and the super user prompt (#).

Take special care when you are logged in as the super user. In particular, you should be careful when deleting or modifying files or directories. This is important because the super user has unlimited access to all files; it is possible to remove or modify a file that is vital to the system. Avoid using wildcard designators in filenames and keep track of your current working directory.

You can leave the super user account at any time by pressing **Ctrl-d**.

Stopping the System

Stopping a UNIX system requires more than just turning off the computer. You must prepare the system for stopping by using either the **shutdown** or the **haltsys** command. The following sections describe each command.

Using the shutdown Command

The **shutdown** command is the normal way to stop the system and should be used whenever the system is in normal operation mode. It warns other users that the system is about to be stopped and gives them an opportunity to finish their work. The warning message that **shutdown** displays at all terminals can be customized. (If desired, the system administrator can also use the **wall**(ADM) command to send a message about the impending shutdown prior to running the actual **shutdown** command.)

To stop the system with the **shutdown**(ADM) command, follow these steps:

1. Log in as the super user. See the section "Logging in as Super User" in this chapter. The system opens the super user account and displays the message of the day and the super user prompt.

2. Enter the following command ands press **Return**:

   /etc/shutdown -g*n*

   Δ **sysadmsh** users select: System→Terminate

   Where *n* is the number of minutes before the shutdown is to take place. If you do not include the **-g** option, you are prompted for the number of minutes. The system displays a warning message at each terminal, asking logged-in users to finish their work and to log out. As soon as all users are logged out or the specified time has elapsed, the system closes all accounts and displays the following message:

```
** Safe to Power Off **
          -or-
** Press Any Key to Reboot **
```

NOTE: If you attempt to select **System→Terminate** from within an X-window, (as opposed to executing **sysadmsh** from the UNIX command line) it will fail. The easiest way to shut down the system is to log in as **root** and use the **shutdown** command.

3. Turn off the computer or press any key to reboot the system.

Using the haltsys Command

The **haltsys**(ADM) command halts the system immediately. This command should be used only when in single-user mode. If there are any users logged into the system when the **haltsys** command is given, they are logged out and their work in progress is lost. In addition, network servers and other programs are terminated abnormally and could create problems when they are restarted.

To stop the system with the **haltsys** command, follow these steps:

1. Log in as the super user. The system opens the super user account and displays the message of the day and the super user prompt.

2. Enter:

   /etc/haltsys

   Now press the **Return** key. The system displays the following message:

```
** Safe to Power Off **
          -or-
** Press Any Key to Reboot **
```

3. Turn off the computer, or press any key to reboot the system.

Understanding the Boot Display Information

At boot time, a table of hardware information is always displayed after the copyright information. This table represents your hardware configuration, as it is recognized by the operating system. Here is an annotated version of the boot screen, as it appears on a sample machine. Comments appear below indented. Figure 3-1 contains an example display.

```
device     address     vector dma comment
-------------------------------------------------------------------------
fpu        -               35   -  type=80387
floppy     0x03F2-0x03F7   06   2  unit=0 type=96ds15
serial     0x02F8-0x02FF   03   -  unit=1 type=Standard nports=1
parallel   0x0378-0x037A   07   -  unit=0
console    -               -    -  unit=ega type=0 12 screens=68k
disk       0x01F0-0x01F7   36   -  type=W0 unit=0 cyls=791 hds=16 secs=48
```


Figure 3-1. Sample Boot Display

This key explains the sample:

`device, address, vector, dma, comment`	The name of the hardware, address in hexadecimal, interrupt vector, direct memory access channel, other details about the hardware.
`fpu`	Floating-Point Unit present, specifically the Intel 80387 chip.
`floppy`	High-Density Floppy Drive.
`serial`	This is COM 1; COM 1 has one port (no multiport card is installed).
`parallel`	This is your parallel port.
`console`	The console has an EGA video adaptor compatible with (type 0) the IBM EGA design.
`disk`	Western Digital st506 controller number 0 (W0), hard drive 0 (unit 0), as well as the number of cylinders, heads, and sectors.

The **hwconfig**(C) utility is used to display or access this information at any time, using the configuration information stored in the file */usr/adm/hwconfig*. Refer to the **hwconfig**(C) manual page for more information.

Chapter 4
Using Filesystems

This chapter describes one of the most important responsibilities of a system administrator: creating and maintaining filesystems. There are four filesystem types that can be used. In addition, general maintenance activities are described, such as strategies for maintaining free space. The concept of "filesystem integrity" is introduced, and how the operating system repairs damaged filesystems. Filesystem creation is discussed in the "Adding Hard Disks" chapter of this Guide.

What Is a Filesystem?

A filesystem is a distinct division of the operating system, consisting of files, directories and the information needed to locate and access them. A filesystem can be thought of as a structure that directories and files are constructed upon.

Each UNIX system has at least one filesystem on the primary hard disk. This filesystem is called "the root filesystem" and is represented by the symbol "/". The root filesystem contains the programs and directories that comprise the operating system. On small hard disks, the root filesystem includes all the user directories as well. The primary hard disk can also be divided into more than one filesystem. One of the most common divisions is the */u* filesystem, used to isolate user accounts from the root filesystem. (For more details on these filesystems, see the *Installation Guide*).

A UNIX system may also have other filesystems that contain special directories and application programs. Dividing the primary hard disk into multiple filesystems is done to protect the data and make maintenance easier. Adding still more filesystems by installing other hard disks expands the system storage space. New filesystems can be specifically created by the system administrator, then "attached" (mounted) and "detached" (unmounted) to the system when needed, the same way that a floppy disk is accessed. The next section describes how to add a new filesystem and, if desired, change the location of user accounts to the new disk. This is done without affecting the present configuration of the primary hard disk. (To change the current organization of filesystems on the primary hard disk, see "Changing/Adding Filesystems on the Primary Hard Disk" in this chapter.

Mounting and Unmounting a Filesystem

The **mount**(ADM) command is used to attach and detach a filesystem. You must specify the type of filesystem you are mounting. For example, to mount or unmount */dev/u* on */u*, you would use the following two commands respectively:

mount /dev/u /u

Δ **sysadmsh** users select: Filesystems→Mount

umount /dev/u

Δ **sysadmsh** users select: Filesystems→Unmount

Only the super-user can use the **mount** command.

NOTE: Files in a filesystem are not accessible unless the filesystem is mounted. If files are copied or created under the mount point while the filesystem is unmounted, those files will appear to be in that filesystem when they are not. When the filesystem is mounted, these files will seem to "disappear" when the filesystem is mounted over them.

Maintaining Free Space in Filesystems

Filesystem maintenance, an important task of the system administrator, keeps the system running smoothly, keeps the filesystems clean, and ensures adequate space for all users. To maintain the filesystems, the system administrator must monitor the free space in each filesystem, and take corrective action whenever the free space gets too low.

This chapter explains the filesystem maintenance commands. These commands report how much space is used, locate seldom-used files, and remove or repair damaged files.

A UNIX system operates best when at least 15% of the space in each filesystem is free. In any system, the amount of free space depends on the size of the disk containing the filesystem and the number of files on the disk. Since every disk has a fixed amount of space, it is important to control the number of files stored on the disk.

If a filesystem has less than 15% free space, system operation usually becomes sluggish. If no free space is available, the system stops any attempts to write to the filesystem. This means that the user's normal work on the computer (creating new files and expanding existing ones) stops.

The only remedy for a filesystem that has less than 15% free space is to delete one or more files from the filesystem. The following sections describe strategies for keeping the free space available.

Strategies for Maintaining Free Space

The system administrator should regularly check the amount of free space on all mounted filesystems and remind users to keep their directories free of unused files. You can remind users by including a reminder in the message of the day file */etc/motd*.

In addition, the **cleantmp**(ADM) command is run by the system to clean the */tmp* directory. You can edit the file */etc/default/cleantmp* to define how often key directories (*/tmp* by default) are cleaned of files. See the **cleantmp**(ADM) man page for details.

If the amount of free space slips below 15%, the system administrator should:

1. Send a system-wide message asking users to remove unused files.
2. Locate exceptionally large directories and files, and send mail to the owners asking them to remove unnecessary files.
3. Reduce disk fragmentation by making a complete backup of the filesystem, removing all the files, and then restoring them again from the backup.
4. If the system is chronically short of free space, it may be necessary to create and mount an additional filesystem.

The above suggestions are described in detail in the following sections.

Displaying Free Space

You can find out how much free space exists in a particular filesystem with the **df** (for "disk free") command. This command displays the number of "blocks" available on the specific filesystem. A block is 512 characters (or bytes) of data.

The **df** command has the form:

df *specialfile*

Δ **sysadmsh** users select: System→Report→Disk

specialfile can be the name of a UNIX special file corresponding to the disk drive containing the filesystem. If you do not give a special filename, then the free space of all normally mounted filesystems is given.

For example, to display the free space of the root filesystem */dev/root*, enter:

df /dev/root

and press **Return**. The command displays the special filename and the number of free blocks. You can find the percentage of free space to total space on your system with the command:

df -v

Displaying Disk Usage

You can display the number of blocks used within a directory by using the **du** command. This command is useful for finding excessively large directories and files.

The **du** command has the form:

du *directory*

The optional *directory* must be the name of a directory in a mounted filesystem. If you do not give a directory name, the command displays the number of blocks in the current directory.

For example, to display the number of blocks used in the directory */usr/johnd*, enter:

du /usr/johnd

and press **Return**. The command displays the name of each file and directory in the */usr/johnd* directory and the number of blocks used.

Displaying Blocks by Owner

You can display a list of users and the number of blocks they own by using the **quot** (for "quota") command. The command has the form:

quot *specialfile*

The *specialfile* must be the name of the special file corresponding to the filesystem.

For example, to display the owners of files in the filesystem on */dev/u*, enter:

quot /dev/u

and press **Return**. The command displays the users who have files in the filesystem and the numbers of blocks in these files.

Removing and Restoring a Filesystem

If your system has been in use for some time, the constant creation and removal of files creates a situation called *disk fragmentation.* This means that the files in the filesystem are written in small pieces on the hard disk. A small amount of disk space is used when a file is written across more than one portion of the disk. You can recover filesystem space (generally about 5 to 10 percent) by first making a complete backup of all the files in the filesystem and then removing all the files from the hard disk and restoring them from the backup. To make a complete backup of your system files, read the "Backing Up Filesystems" chapter in this Guide for instructions on how to backup and restore a filesystem.

Because the files are completely rewritten on the disk, each file is written in one piece and fragmentation is reduced. A small amount of space will be recovered. It is a good idea to perform this action about once a year on a heavily used system and less often on a lightly used system. Be certain that you have complete, accurate, and readable backups before you begin or your files may be lost.

Filesystem Integrity

We have explained that a filesystem is a distinct division of the operating system. It is part of the job of the operating system to maintain the integrity of filesystem data. Actual loss of data is a rare occurrence; UNIX filesystems are very resistant to corruption. This is because a certain amount of redundancy exists in special structures that are invisible to the user. It is these structures that ensure filesystem integrity. For example, when the system suffers a power outage, very little information is lost. Any damage usually affects one or two files, making them inaccessible. In almost all cases, the operating system can repair any damage done to files. In very rare cases, damage causes the entire filesystem to become inaccessible.

The operating system uses the **fsck** program to repair damaged filesystems. **fsck** (for "filesystem check") checks the consistency of filesystems. In cases where the contents of a file are lost (rare), the only way to restore lost data is from filesystem backups. **fsck** is run automatically on the root filesystem at boot time. The **fsck** status messages look like this:

```
** Phase 1 - Check Blocks and Sizes
** Phase 2 - Pathnames
** Phase 3 - Connectivity
** Phase 4 - Reference Counts
** Phase 5 - Check Free List
```

If the system terminated abnormally (power outage), you see other messages that may seem alarming:

```
FREE INODE COUNT WRONG IN SUPERBLK (FIX?)
```

In fact, this kind of message is routine when a system was not shut down properly, and you have only to enter **y** and **fsck** will continue its recovery work. This could be done without the system administrator's intervention, but it is generally better to know what is happening to a filesystem after a problem has occurred.

In order to discuss the idea of filesystem integrity and how **fsck** functions, it is useful to explain the structure that underlies the simple idea of files, directories, and filesystems. Although it is not necessarily vital to understand the principles of file storage, it is helpful to know what the messages like the one above refer to so they will seem less mysterious. After reading this section, you will understand some of the most basic principles of UNIX operating systems. The section "Repairing a Filesystem with **fsck**" explains the simple mechanics of using the **fsck** command. The subsections that follow explain the filesystem structures that **fsck** deals with.

How UNIX Systems Maintain Files

Each filesystem contains special structures that allow the operating system to access and maintain the files and data stored on the filesystem. It is the disruption and repair of these structures that we are concerned with.

The structure of a filesystem is based on the way that hard disks store data. Although the hard disk contains all the data used by the system, it is not stored in neat little locations that correspond to individual files. It is unlikely that you could point to a spot on a hard disk and truthfully say: "My file is stored right there on this part of the disk." In fact, the data is probably scattered across the disk, and a sophisticated addressing scheme is used by the operating system so that it can access each of the pieces that a file is broken into and present it to the user as a unit.

The data is spread around because the operating system doesn't really deal with files, but with units of data. To explain what this means, let's assume that a file is created and is actually stored on one part of the disk. Now, suppose you edit that file, and delete a few sentences here and there. That means you are using a little bit less disk space than when you started. This space amounts to a series of gaps in the area where your file was stored. Disk space is a precious commodity and is not wasted. Those small amounts of memory are then allocated to other files. Picture this process on a scale of hundreds of files with a dozen users and you have an idea of how files are maintained. Because of the effectiveness of the algorithms (formulas) that the operating system uses, this process is remarkably efficient and trustworthy.

How UNIX Systems Maintain Filesystems

A filesystem contains files and directories, which are represented by special structures called "inodes" and "data blocks" that make it possible for the operating system to create and keep track of them:

Data blocks A block is a 1024-byte unit of data stored on the disk. A data block can contain either directory entries or file data. A directory entry consists of an inode number and a filename.

Inodes Inodes can be thought of as a card from a library card catalog. Each inode contains information about a file, just as a card contains information about a book, including its location, its size, the type of file, and the number of directory entries linked to the file. One important point to remember is that an inode does not contain the name of a file; directories contain the actual names. Inodes contain the locations of all the data that makes up a file so the operating system can collect it all when needed.

Blocks are not just stored on the hard disk. In order to minimize seeking data on the hard disk, recently used data blocks are held in a cache of special memory structures called buffers. It is these structures that make the operating system more efficient. When enough data is accumulated to write out one or more full disk blocks, the buffer is "flushed" by writing its information to the disk. Some information is always lost when an outage occurs because recently changed data that has not yet been written to the disk, but this is very minor.

With a hard disk filled with data, inodes, directories, files, and blocks cached in memory, how does the operating system keep track of them? The answer is that all these structures maintain sufficient connectivity between files and directories to allow severed connections to be reconstructed.

One special data block, the "super" block, contains overall information about the filesystem, rather than where a particular piece of a file is located. The super block contains the information necessary to mount a filesystem and access its data. It contains the size of the filesystem, the number of free inodes, and information about free space available.

Information is read from the disk version of the super block when the filesystem is mounted and is maintained and modified in memory as activity takes place on the system. The information is written back to the disk at regular intervals by the **update** command, which is normally executed by the **/etc/rc2** scripts when the system is brought up. The **update** command calls the **sync**(C) command every 30 seconds, which forces the memory version of the super block and buffers to be written to disk. If the system crashes and the information stored on the disk is not reasonably up-to-date, the filesystem might be corrupted.

Causes of Filesystem Corruption

In the case of all the structures mentioned in this section, corruption can occur. This means that the data or the structures used to locate data can be damaged. This can occur for several reasons:

Hardware Failure	Hardware failures are rare. The best way of dealing with it is to be sure that recommended diagnostic and maintenance procedures are followed conscientiously.
Program Interrupts	It is possible that errors that cause a program to fail might result in the loss of some data. It is not easy to generalize about this because the range of possibilities is so large.

Human Error — While it may be painful to admit it, probably the greatest cause of filesystem corruption falls under this heading. There are rules that should be followed when managing filesystems.

Rules for Checking Filesystems

1. ALWAYS check a filesystem with **fsck** before mounting it. Nothing complicates the problem of cleaning up a corrupt filesystem so much as allowing it to be used when it is bad.
2. NEVER remove a filesystem physically without first unmounting it.
3. ALWAYS use the **sync** command before shutting the system down and before unmounting a filesystem. (The **sync** command writes data in the buffer cache back to the disk.)

Regular filesystem backups represent the best assurance of continued filesystem integrity.

Repairing a Filesystem with fsck

You can repair a filesystem with the **fsck** command. A filesystem must be unmounted before running **fsck**. (The root filesystem can't be unmounted, so the system must be shut down first and brought up again in single-user (maintenance) mode.) **fsck** examines the various structures on the disk and attempts to reconcile them. Where possible, it reestablishes connections, resolves references, it "cleans" a filesystem.

The command has the form:

fsck *specialfile*

Δ **sysadmsh** users select: Filesystems→Check

The *specialfile* must be the name of the special file corresponding to the device name of the filesystem.

NOTE: The **fsck** program is actually a front-end that invokes a version of **fsck** for each filesystem type. **fsck** calls a special version to repair DOS filesystems.

For example, let's assume that you have brought up your system after a power failure and are in single-user mode. To check the filesystem */u*, which is represented by the device */dev/u*, you would enter:

fsck /dev/u

and press **Return**. The program checks the filesystem and reports on its progress with the following messages:

```
** Phase 1 - Check Blocks and Sizes
** Phase 2 - Pathnames
** Phase 3 - Connectivity
** Phase 4 - Reference Counts
** Phase 5 - Check Free List
```

If a damaged file is found during any one of these phases, the command asks if it should be repaired or salvaged. Enter **y** to repair a damaged file. You should always allow the system to repair damaged files even if you have copies of the files elsewhere or intend to delete the damaged files.

Note that the **fsck** command deletes any file that it considers too damaged to be repaired. You can elect to have **fsck** either make the repair or not. The reason you might choose to have **fsck** ignore an inconsistency is that you judge the problem to be so severe that either you want to fix it yourself using the **fsdb(ADM)** utility, or you plan to restore your system from backups. If you cannot use **fsdb**, you must allow **fsck** to resolve the inconsistencies or the filesystem may not be usable.

Note that you may need to run **fsck** several times before the entire filesystem is clean. For a complete list of error messages, see the **fsck**(ADM) manual page.

Summary of fsck Phases

fsck scans and examines each of the structures mentioned earlier. Each phase compares components and checks that these components agree with each other.

Phase 1 checks the blocks and sizes. **fsck** reads the inode list to determine the sizes and locate the blocks used by each file. Inodes are checked for validity which includes checking inode type, zero link counts, inode sizes, and for bad or duplicate blocks. (Bad blocks are block values outside the boundary of a filesystem.) When **fsck** asks whether or not to clear an inode, this means to zero out the bad information in the node. This removes the file or directory that was associated with it. A duplicate block means that two inodes point to the same block on the disk. **fsck** attempts to find the original inode along with the duplicate for correction in Phase 2.

Phase 2 checks the path names. Files removed in Phase 1 must then have their directory entries removed. Phase 2 cleans up error conditions caused by improper inode status, out-of-range inode pointers, and directories that point to bad inodes as described earlier. For files with duplicate blocks found in Phase 1, **fsck** will want to remove both files (this is one of the few areas where system administrator intervention is useful).

Phase 3 checks for connectivity. Phase 2 removed directories that don't point to valid files. Phase 3 reconnects files that have been severed from the directory structure. Any files that are unreferenced but valid are placed in a special directory called *lost+found.* Because the directory has been severed, the name of the file is lost and a number is assigned to the file in *lost+found.*

Phase 4 checks the reference counts. **fsck** checks the link count of each entry that survives Phases 2 and 3. In some cases, files that were not pointed to under the directory structure but still have an inode can be relinked back to the filesystem in *lost+found.*

Phase 5 checks the free list. **fsck** examines the free-block list maintained by the filesystem and resolves the missing or unallocated blocks allocated or removed earlier. When an inconsistency is detected, **fsck** prompts to rebuild it.

Phase 6 salvages the free list. If specified in Phase 5, the system reconstructs a free-block-list from the altered filesystem.

Automatic Filesystem Check

The operating system sometimes requests a check of the filesystem when you first start it. This usually occurs after an improper shutdown (for example, after a power loss). The filesystem check repairs any files disrupted during the shutdown.

Chapter 5

Maintaining System Security

Every computer system needs protection from unauthorized people accessing the computer, disks and system files. The security features present on your system represent enhancements to the basic security features of UNIX operating systems. The operating system is designed to meet the requirements of the C2 class of trust as defined by the Department of Defense's *Trusted Computer System Evaluation Criteria* (or, the *Orange Book*).

This chapter shows you how to use the security features to maintain a trusted system. Features affecting the ordinary user are described in *Using ODT-OS* in the *Administrator's Guide*.

This chapter includes the following information:

- An overview of system security
- A description of protected subsystems
- How to assign administrative roles
- Administering subsystems with the **sysadmsh**
- Using the audit subsystem
- Protecting filesystems
- Checking the integrity of the system
- Error messages
- Adding dial-in password protection

NOTE: About Physical Security

The security features of the operating system are useless if your hardware and media are not protected. You must protect the computer itself, the distribution diskettes, and any backup media from unauthorized access. This is accomplished by the following rules:

1. Keep your system under lock and key when an operator is not present.

2. Organize and lock up all backup media.

What Is a Trusted System?

Because there is no such thing as a computer system that is completely free from risk, systems are referred to as "trusted" rather than "secure." This is what is meant by the C2 class of "trust." A "trusted" system is one which achieves a specific level of control over access to information, providing mechanisms to prevent (or at least detect) unauthorized access.

The security features of the operating system are an extension of features present on typical, less "trusted" UNIX systems. Full compatibility with existing UNIX mechanisms is maintained while expanding the protection of user and system information. A large part of system administration involves maintaining and protecting system information as described in this section.

The system can be configured at installation time to operate in a manner consistent with most UNIX systems, in a non-trusted state with few restrictions in place beyond the normal UNIX mechanisms. (To change the security defaults to C2 after installing with relaxed defaults, refer to "Selecting the C2 Security Defaults" under "Default Account Configuration" in the "Administering User Accounts" chapter of this Guide.) When the C2 security parameters are selected and auditing is enabled, the system is configured to operate in a "trusted" state. In this state, only a designated administrator can access or modify system information. Once the system goes into normal operation, the system's trusted state must be maintained if you wish to take full advantage of the trusted features. If you follow these guidelines, system information remains protected.

As administrator, your actions are crucial to maintaining a trusted system. Any lapses from a trusted state invite system penetrations. To be effective in your administrative position, you must understand the system's security policy, how it is controlled by system information (databases), and how changes you make affect user and administrator actions.

Trusted System Concepts

The following defines the basic concepts of a trusted system. As administrator, you must understand these concepts and know where security-relevant information is kept to run the system properly. This section only introduces these topics; later sections in this chapter provide details and maintenance procedures.

Trusted Computing Base

A collection of software called the **Trusted Computing Base** (TCB) maintains the parts of the system's state that are related to security. The TCB consists of the UNIX kernel and the trusted utilities that reference and maintain relevant security data. (The *kernel* is the heart of the operating system.) The Trusted Computing Base implements the security policy of the system. It oversees and guards interactions between *subjects* (such as processes) and *objects* (such as files, devices, and interprocess communication objects). Much of the software that you interact with is part of the system's TCB. The **sysadmsh**(ADM) provides a menu-driven, administrative interface to help you maintain the Trusted Computing Base.

Accountability

An action is *accountable* if it can be traced to a real user. In a secure system, all users are responsible for their own actions, and all actions can be traced to a responsible user. Most UNIX systems lack good accountability because some actions cannot be traced to any user. For example, pseudo-user accounts, such as **lp** or **cron**, run anonymously; their actions can be discovered only by changes to system information. As described later, a trusted UNIX system improves accountability by associating each account with a real user, auditing every action, and associating each action with a specific user on the system.

On a typical UNIX system, each process has a real and effective user ID as well as a real and effective group ID. A process with the effective user ID set to **root** can set these identifiers to any user. The concept of user identity has been expanded to add a separate identifier called the *login user identifier* (LUID). The LUID is an indelible stamp on every process associated with a user. The LUID identifies the user who is responsible for the process's session. Once stamped, the process's LUID cannot be changed by anyone. Child processes inherit the LUID of their parent.

Discretionary Access Control

Discretionary access control determines whether a user has access to the information that they wish. That information is within an *object* (file, device, etc.) that the user's *process* is trying to use. On most UNIX systems, object protection is enforced through the relationship between the user and group of a process and the owner, group and other mode bits of the object. The protection attributes of these objects are at the *discretion* of the object's owner. Therefore the protection attributes can be changed by the owner to be restrictive (controlled access) or permissive (open access). A trusted UNIX system extends the standard discretionary access control rules to enforce greater protection over system information (system databases), shared information (temporary directories), and SUID (set user ID on execution) program input files (for example, mail messages).

In addition, a mechanism has been added to the UNIX discretionary access policy that can prevent an SUID program from gaining access to the invoking user's files. A user can create a *promain*, or *pro*tected do*main*. A promain is a directory hierarchy (subtree). When the user invokes an SUID program, the program can access their private files only if those files are within that subtree. Outside the promain, the SUID program can access only those files that can be accessed by both the program's invoker and the program's owner. This limits the damage an SUID program might inflict in a given directory. This mechanism is discussed in detail in **promain**(M).

Authorizations

An *authorization* is a right to access some object or perform some system action. Most UNIX systems make all access decisions based on the simple discretionary considerations or on whether the process making the access is owned by **root**. The root account has the power to perform system actions that no other process can. The Operating System defines two types of authorizations: kernel authorizations and subsystem authorizations. *Kernel authorizations* are associated with processes. (A process is a program that is currently running on a system.) They allow a process to perform certain actions if the process has the requisite privilege. *Subsystem authorizations* are associated with users. They allow the user to perform a special action using a subsystem's commands (trusted utilities and programs). A *subsystem* is a related collection of files, devices, and commands that serve a particular function. For example, the **lp** subsystem consists of the print spooler files, the printer devices, and commands such as **lpadmin**(ADM) that help maintain the subsystem.

Kernel authorizations are stored in an *authorization set* associated with every process. The authorization set is a list of privileges that allow a type of action if the privilege is present, and do not allow the action if the privilege is absent. Authorizations are discussed later.

Identification and Authentication (I&A)

When a user logs into a less-trusted UNIX system, limited identification and authentication takes place. The system searches the password database (**/etc/passwd**) for the user name. If the user name is found, the system authenticates the user by comparing the password entered to the encrypted version of the password in the user's password database entry. Some rules concerning the characteristics of the password and the ability to change it are enforced, but these rules have been shown to be insufficient to guard against penetration.

A trusted system extends the standard I&A mechanisms. There are more rules enforcing the types of passwords that can be used. There are new procedures for generating and changing passwords. The location and protection of certain parts of the password database have been changed. And the administrator has greater control over user actions. A separate role, called *Authentication Administrator* (subsystem authorization **auth**), has been created to maintain this aspect of the system. That administrator's responsibilities are described in detail in later sections.

Audit

Most UNIX systems keep a limited record of system actions with its accounting subsystem. The accounting subsystem writes a single accounting record upon completion of each user process. The trusted operating system provides an extensive series of records, or *trail*, of actions. In this trail is a record of every access between subject and object, and every change of subject, object, and system characteristics. The audit subsystem is controlled by a separate role called *Audit Administrator* (subsystem authorization **audit**), The audit administrator decides how much information is recorded, decides how reliably it is recorded, and maintains the information once it is collected. The audit subsystem provides the audit administrator with an extensive history of system actions. This helps the administrator to identify what happened to the system when and who was involved.

Protected Subsystems

UNIX systems provide the *setuid* (SUID) and *setgid* (SGID) mechanisms. With these you can construct programs maintaining private information. This information can only be accessed or modified by the operations implemented in the programs. The Operating System defines several *protected subsystems*. Each of these subsystems consists of a collection of private information (files and/or databases), any related devices, and the utilities and commands used to maintain that information. The protected subsystems use the SUID/SGID mechanisms to protect their private files, databases and devices from unrestricted access. The Operating System extends the notion of a protected subsystem in several ways:

- It provides more precise control of users and groups that maintain certain collections of system resources (private information).
- It keeps a separate database of users allowed to run the programs that maintain the private information.
- It does not require users to log in as the subsystem administrator but rather uses the database to check the subsystem authorization. This satisfies the full accountability requirement for all actions performed by subsystem programs.

Running a Trusted System

This section discusses the framework for maintaining your trusted system. The first basic choice you must make is who will maintain it. You can have a single, all-powerful superuser with the **root** login, or you can assign parts of the administrative responsibility to other users, assigning no more power than is necessary to administer a single aspect of system operation.

Assigning Administrative Roles via Authorizations

The administrative tasks for a trusted UNIX system are split into a number of logical *roles*. Each role is responsible for maintaining one aspect of the system. The idea of specific administrative roles (and their associated tasks and responsibilities) is pivotal to your understanding of a trusted operating system. Each logical role can be assigned to the same person or to separate members of an administrative staff. Each extended role has a special authorization associated with it. That association, together with a sophisticated tracking system, enables the administrator to maintain a clear record of administrative actions. This helps prevent problems and makes other problems easier to identify and solve.

In order to perform the tasks associated with an administrative role, an administrator must have the appropriate authorization. Table 5.1 lists the administrative roles, associated authorizations, and the areas of the system maintained by each role.

Table 5.1.

Protected Subsystems and Administrative Roles

Role	Subsystem Authorization	Area
Authentication Administrator	auth	System Accounts
Printer Administrator	lp	Line printer subsystem
Terminal Administrator*	terminal	Terminal device permissions
Cron Administrator*	cron	**at** and **cron** subsystem
Memory Administrator*	mem	Access to system memory
Audit Administrator	audit	Audit databases and audit trail
Operator	backup	File system backups
System Administrator	su	su access to super-user account
	sysadmin	Use of **integrity**(ADM) program

* These are not really administrative roles, as will be explained later.

It is vital that you understand the responsibilities for each role and the impact of your actions on the security of the system. You should configure and run the system based on the sensitivity of information kept on your site, the perceived degree of cooperation and expertise of your users, and the threat of penetration or misuse from insiders and outsiders. Only your vigilance and proper use of the system can keep the system trusted and protect the integrity of your system.

To assign a subsystem authorization, return to the desktop and double-click on the Sysadmsh icon to bring up the main **sysadmsh**(ADM) menu. Then make the following selection:

Accounts→User→Examine:Privileges

NOTE: You might notice that each subsystem authorization appears to be identical to the group name for that subsystem. This means that if a user is a member of a subsystem group, there is an implied ability to access the files of that subsystem. However, this does not confer the requisite subsystem authorization. In fact, you should never make a user a member of a subsystem's group, as this can put actual data files at risk. Use the proper subsystem authorization to permit access to the subsystem.

Administering Subsystems with the sysadmsh

Certain subsystems are logical divisions rather than actual areas of system administration. For example, the Memory subsystem isn't administered per se, but the assignment of the **mem** authorization controls access to kernel memory structures. Other subsystems require administration and have **sysadmsh**(ADM) selections. These subsystems can be assigned to individuals and documentation is provided for each area. Table 5.2 lists each of the subsystems that must be administered, their **sysadmsh** selections and the chapters/sections that deal with them.

Table 5.2.

Subsystems and sysadmsh Selections

Protected Subsystem	sysadmsh Selection	Chapter or Section
Line Printer	Printers	"Using Printers"
Backup	Backups	"Backing Up Filesystems"
Authentication	Accounts	"Administering User Accounts"
Cron	Jobs	"Authorizing the Use of Job Scheduling Commands" (in this chapter)
Audit	System→Audit	"Using the Audit Subsystem" (in this chapter)

The **audit** subsystem is described later in the chapter. The subsystems are described in detail in **subsystem**(M). This page lists all the programs and data files associated with a subsystem. Most of the functionality normally exercised by the super-user on other, less-trusted UNIX systems have been delegated to the protected subsystems detailed in this section. However, some functions still need to be done by the super-user. This includes mounting and unmounting filesystems, and traversing the entire file tree. Only the super-user can do everything. Users having the **su** authorization can **su**(C) to the super-user account. Restrict the **root** password to a few users and assign a responsible user to the **root** account. (See the "Administering User Accounts" chapter of this Guide for details.)

Assigning Kernel Authorizations

The operating system has two types of authorizations: kernel and subsystem. User processes run with a set of kernel authorizations that control the special rights the process has for certain privileged system actions. Table 5.3 contains is a list of kernel authorizations.

Table 5.3.

Kernel Authorizations

Authorization	Action
configaudit	Modify audit parameters
writeaudit	Write audit records to the audit trail
execsuid	Run SUID programs
chmodsugid	Ability to set the SUID and SGID bit on files
chown	Change the owner of an object
suspendaudit	Suspend the audit of a process
nopromain	Access as user outside the promain directory

Most users require only **nopromain** and **execsuid** kernel authorizations to perform routine tasks. (Promains are used to isolate the execution of a given program rather than as a ongoing protection measure. **nopromain** is the usual mode that a user would operate with, temporarily deassigning it to test the execution of a program. See **promain**(M) for more information). In order to execute SUID programs, a user must have the **execsuid** authorization. (Set-user-ID programs run under an ID other than that of the invoker. This is done so that the process can access files and system facilities that would otherwise be inaccessible.) If the user needs to create files with the SUID or SGID bits, they must have the **chmodsugid** authorization. In order to change ownership of a file (give it away) the **chown** authorization is required. If a user does not have this authorization, ownership of files can only be changed by **root**.

NOTE: Restricted *chown* is required for NIST FIPS 151-1 conformance. The **chown** authorization should not be assigned to users if you wish to conform to NIST FIPS 151-1 requirements.

Each process has an *authorization set* listing its kernel authorizations. A process can only perform a specific action if the appropriate kernel authorization is listed in the set. For example, a process with **configaudit** authorization can modify the parameters of the audit subsystem. Such a process is run by the administrator whose role is *Audit Administrator*. No user processes should run with any of the audit authorizations. These are intended for use

only by the Audit Administrator. To assign a kernel authorization, return to the desktop and double-click on the Sysadmsh icon to bring up the main **sysadmsh**(ADM) menu. Then make the following selection:

Accounts→User→Examine:Privileges

Users assigned administrative roles must have certain kernel authorizations in order to perform the tasks required by the subsystem. If you use the "C2" or "Relaxed" system-wide default kernel authorizations, you need only assign the necessary audit authorizations; **chown**, **execsuid**, and **chown** are already assigned.

Table 5.4.

Subsystem Kernel Authorization Requirements

Subsystem	Required Kernel Authorization
audit	configaudit, suspendaudit, execsuid
auth	chown, execsuid
backup	execsuid
lp	chown
cron	execsuid, chown, chmodsugid
sysadmin	execsuid, chmodsugid, chown

Using Customized or Default Security Parameters

As discussed earlier, there are two sets of defaults available: C2 and "Relaxed", which corresponds to the behavior of less-trusted UNIX systems. In addition, each parameter can be changed for individuals as well as system-wide. To change the system-wide parameters, see "Default Account Configuration" in the "Administering User Accounts" chapter. See "Account Management" in the same chapter for information regarding parameters for individuals.

Controlling System Access

One important aspect of operation on a trusted system is locating potential problems relating to security. The restriction mechanisms fall into three categories, all of which can be customized and reported on:

- Password status
- Terminal activity
- Login activity

Each of these areas is important to pinpointing potential security trouble. The procedures for running these reports are found in "Activity Report Generation" in the "Administering User Accounts" chapter. The subsections that follow explain how these restrictions are implemented.

Password Status

The Department of Defense *Password Management Guideline* has been used as a model for password restrictions, and users are subject to much stricter password checking than less-trusted UNIX systems. The Authentication Administrator has the choice of allowing users to pick their own passwords or have the system generate passwords for them. Once chosen, the password can be subjected to much more extensive triviality checks, again at the option of the Authentication Administrator.

Passwords have three levels of validity, extending the password aging implementation in standard UNIX. Passwords are *valid* until their *expiration time* is reached. When a password expires, if the user is allowed to do so, the user is prompted to change their password. If users are not permitted to change their own password, you, the administrator, will have to assign a new password.

Passwords are considered *expired* until their *lifetime* ends. A *dead* password causes a user account to be locked. Only the Authentication Administrator can remove the lock on the user's account, through the **sysadmsh** Accounts selection. If users are not allowed to change passwords, a new password must be assigned.

To prevent password reuse, the Authentication Administrator can also set a *minimum change time* on a password, before which a user may not change passwords. *All* of these parameters can be changed on a system-wide (System Defaults database) and per-user (Protected Password database) basis. See "Changing Default Password Restrictions" in the "Administering User Accounts" chapter of this Guide for changing system-wide password parameters, and "Changing a User Password or Password Parameters" in the same chapter for a full discussion of password changing procedures.

The password reports can be used to keep users from getting their accounts locked, an annoying occurrence if a system administrator is unavailable. If your system is not attended by administrators on a daily basis, you might want to extend the password lifetime parameter accordingly.

Login Activity

User accounts have parameters associated with the number of login attempts and retry interval. for accounts rather than terminals.) To change or examine login restrictions, refer to "Changing Default Login Restrictions" in the "Administering User Accounts" chapter of this Guide.

Creating an Override Login

You should create an override login entry for root in case the security databases become corrupted and all logins are disallowed. This is done by adding a special entry to the file */etc/default/login*. This identifies the tty to be used when doing an override login for root. For example, the following entry permits root to log in on tty01, also known as the first multiscreen.

```
OVERRIDE=tty01
```

When **root** logs in on the override tty when the databases have been compromised, the following message is generated:

```
The security databases are corrupt.
However, login at terminal tty is allowed.
```

When **root** logs in on the override tty when the account has been locked, the following message is generated:

```
Account is disabled but console login is allowed.
```

Using the Audit Subsystem

The Audit Subsystem records security-related events that occur on a system in the form of an *audit trail* that can later be examined. Audit trails produced by this subsystem can detect penetration of the system and the misuse of resources. The Audit Subsystem is designed to meet the audit goals specified by the U.S. National Computer Security Center.

Audit permits the review of the collected data to examine patterns of access to *objects* and to observe the actions of specific users and their processes. Attempts to violate protection and authorization mechanisms are audited. The audit subsystem provides a high degree of assurance that attempts to bypass security mechanisms are audited. Since security-related events are audited and are accountable to a specific user, the audit subsystem serves as a deterrent to users attempting to misuse the system.

The audit subsystem uses system call and utility usage to classify user actions into event types. These can be used for selective audit generation and reduction. One such event type, *Discretionary Access* (DAC) *Denial*, records attempts to use objects in a manner not permitted by the object's permissions. For example, if a user process attempts to write a read-only file, a DAC Denial event is audited, showing that the process tried to write a file to which it was not entitled. When you examine the audit trail, it is easy to notice repeated attempts to access files for which permission is not granted.

Essential to the effectiveness of the audit data is the accountability of users performing the actions being audited. As users attempt to log onto the system, they must go through an identification and authentication process before access to the system is granted. The security mechanism stamps each process created by a user with an immutable indicator of identity: the Login UID or LUID. The LUID is preserved regardless of transitions between user accounts with commands like **su**(C). Each audit record generated by the subsystem contains the LUID along with the process's effective and real user and group IDs. As a result, users can be held strictly accountable for their actions.

The Audit Subsystem is administered by the Audit Administrator. As the Audit Administrator, you have complete control over the events selected for audit record generation, over the parameter values for subsystem control, and over the subsequent reduction and analysis of audit data.

Audit Subsystem Components

The Audit Subsystem consists of five major components:

- Kernel audit mechanism
- Audit device driver (*/dev/audit*)
- Audit compaction daemon - **auditd**(ADM)
- **sysadmsh** audit interface
- Data reduction/analysis facility.

Although not actually part of the audit subsystem proper, there are a number of trusted system utilities responsible for writing audit records to the audit trail (such as **login**(M)).

Kernel Audit Mechanism

The kernel audit mechanism is central to the audit subsystem. This mechanism generates audit records based on user process activity through kernel system calls. Each kernel system call contains an entry in a subsystem table that indicates whether the call is security-relevant and, if so, to what event type the system call corresponds. Additionally, a table of error codes is used to further classify the system calls into specific security events. The kernel audit mechanism makes an internal call to the device driver to write a record to the audit trail.

For instance, the **open**(S) system call is classified as a *Make object available* event. If user **blf** performs an **open**(S) on */unix* and it succeeds, an audit record is generated indicating the event. If however, the system call fails because **blf** requested write access on the **open**(S) but does not have write permission on the file, the action is classified as a *Discretionary Access Denial* event for **blf** with object */unix*. Consequently, a system call may map to a number of event types, depending on the object accessed and/or the result of the call. It is therefore possible that a system call might be audited selectively, depending on the event types that you enable.

Some system calls are not considered relevant to security. For instance **getpid**(S), retrieves the process ID for a process and does not constitute an event of security relevance. Thus, that system call is never audited.

Audit Device Driver

The audit device driver is responsible for: accepting audit records from the kernel audit mechanism and from trusted utilities; creating and writing the intermediate audit trail files; providing audit trail data to the audit daemon for compaction; and, providing for selective audit record generation based on event types, user IDs, and group IDs.

The device driver provides **open**(S), **close**(S), **read**(S), **write**(S), and **ioctl**(S) interfaces like many other character devices. However, the audit device may only be opened by processes which have **configaudit** and/or **writeaudit** kernel authorizations. This limits access to the audit device only to trusted utilities such as the audit daemon and the Audit Administrator interfaces. The audit device may be written to by many processes at the same time. The device handles the merging of the records into the audit trail. The device may only be read by a single process, the audit daemon.

The audit device driver maintains the audit trail as a set of *audit collection files*. Each time you enable auditing, a new audit session is begun. As the session starts, the subsystem creates a collection file into which audit records are written. When the collection file reaches a certain size, specified by the administrator, the subsystem creates a new collection file and begins writing to it. The audit trail could therefore be viewed as a continuously growing

sequential file even though many collection files are used. That is precisely how the audit trail is viewed by the audit daemon because it reads the device and is presented records from the audit trail. The subsystem handles the necessary switching to new collection files for the daemon when the end of a file is reached. All of this is transparent to the daemon.

Audit Compaction Daemon

The audit daemon **auditd**(ADM) is a trusted utility that runs as a background daemon process whenever you enable auditing. The daemon is the sole reader of the audit device which, in turn, provides the daemon with blocks of records from the audit collection files. The daemon is not concerned that the audit trail is spread over numerous collection files. The audit device driver satisfies the read requests from the daemon and handles the switching and deletion of collection files as needed.

The main purpose of the daemon is to provide a compaction and logging mechanism for the audit session. The daemon also serves a support role for protected subsystems, enabling them to write audit records to the subsystem. Depending on the audit record generation criteria you select, a large amount of audit data can be generated on the system. For a typical single-user system, it would not be uncommon to generate 200 Kbytes of audit data in an hour. The daemon therefore provides a compaction mechanism, compressing the audit data into a packed record format that is stored in an *audit compaction file*. The compaction algorithm provides for an average 60-percent reduction in file space. This greatly reduces disk space usage by audit records.

A second function of the daemon is to provide a log file describing the current audit session. The log file contains information about: the number of audit records available in the compacted files output for the session; the start and stop times of the session; and other indicators pertaining to the audit session's state. Just as the audit device driver switches collections files as they reach administrator-specified sizes, the daemon may create multiple compaction files to avoid growing a single file too large to be manageable. You also control this. Audit compaction files written by the daemon may also be located in a variety of administrator-specified directories. For these reasons, the log file is maintained to provide a trail of compaction files that may be used for subsequent data reduction.

A third function of the audit daemon is to serve as an interface program to the audit device driver for the writing of audit records from protected subsystems which do not have the **writeaudit** authorization. Since these subsystems cannot access the audit device driver directly but can interface to the daemon in a secure manner, the daemon handles the writing of the application audit record to the subsystem.

sysadmsh Audit Selection

sysadmsh presents simple options to set up and maintain the audit subsystem. This allows the administrator to handle set up and initialization, modify subsystem parameters, maintain the subsystem (backup, restore, and so on), and reduce both general and selective audit data.

Data Reduction/Analysis Facility

sysadmsh also includes a data reduction/analysis utility to facilitate the examination of audit trails from previous audit sessions or from the current audit session. By using the log file produced by the audit daemon, the reduction utility is able to identify all of the compaction files needed to reduce an audit session. Since the compaction files are in a compressed format, the reduction program contains the necessary routines to uncompress the data.

To provide effective analysis of audit data, the reduction utility lets you specify certain event types, user IDs, group IDs, and object names to selectively reduce the data. Furthermore, you can specify a time interval to be applied while searching for records to match the specified criteria. If a record is not within the specified time interval, it is discarded for the purpose of that reduction.

As an example, you may reduce the data selecting the DAC Denial event with user ID **blf** looking for the object */unix*. Only records that reflect an access attempt to */unix* by **blf** that was denied because of lack of permission are printed. This provides a powerful mechanism for identifying security events of immediate interest without having to analyze the entire audit trail.

Audit Methodology

This section explains how the audit subsystem functions, what criteria are used to collect data, and how audit requirements affect system performance.

Audit Authorizations

There are three authorizations associated with the audit subsystem: configaudit, writeaudit and suspendaudit.

The **configaudit** authorization allows the audit parameters for all users of the system to be set.

The **writeaudit** authorization allows specific information to be recorded by the audit trail.

The **suspendaudit** authorization prevents any auditing.

Audit Record Sources

The audit trail contains the security-related events for the system. Effective auditing concerns not only system call requests from user processes but also certain events such as login, logoff, and login failure attempts. These events are critical to determining who has accessed the system, at what times, from what terminal, and what actions were performed. Login failures are impossible to audit at the kernel level since the kernel has no knowledge of what an application is specifically doing. Thus, certain security-critical utilities such as **login** must be allowed to generate audit records.

Audit records are generated from three sources (discussed in the sections that follow):

- Kernel audit mechanism
- Trusted application processes
- Authorized subsystems

Kernel Audit Mechanism

A large percentage of the audit records stored in the audit trail are generated by the kernel audit mechanism. This portion of the audit subsystem generates records in response to user process system calls that map to security-related events. Some system calls, **open**(S) for instance, map to multiple security events depending upon user arguments and the state of the file being opened. If **open**(S) is called with the **O_CREAT** flag, the file is created if it does not exist. If the **O_TRUNC** flag is specified, the file is truncated to zero length if it exists. This illustrates how the **open**(S) call could map to one of three distinct events, *Make Object Available*, *Object Creation*, or *Object Modification.*

Error codes also play an important role in determining the event. Errors on system calls that indicate access or permission denials as well as resource consumption problems are mapped to specific event types. The kernel audit mechanism determines at the end of the system call what event class the call belongs to and if that event is to be audited as specified by you. Furthermore, the mechanism may also apply additional selection criteria such as user ID or group ID. In this manner, the generation of audit records can be limited to a select group of users.

Trusted Applications

The Trusted Computing Base (TCB) contains a number of trusted applications essential to providing a secure environment. Among these are **login**, **su**, and various audit subsystem commands. To reduce the amount of audit data written to the audit trail, and to make the trail more meaningful, these trusted applications are permitted to write directly to the audit device. This enables **login**, for instance, to write a login audit record to the audit trail rather than letting a login on the system be represented as a collection of system calls required to complete the login procedure.

It is not sufficient to just let the trusted applications write to the audit device. There must also be a way to suppress the generation of system call audit records by the kernel audit mechanism to avoid the problem of a cluttered audit trail. Thus, the **suspendaudit** authorization exists as discussed earlier. Trusted applications run with this authorization enabled, suspending kernel system call auditing for that process and allowing it to open and write the audit device. Only a few trusted applications are permitted to do this. A normal user process never runs with **suspendaudit** authorization. The authorization mechanism is managed by **login**, using restricted system calls, and is based on Protected Password database entries.

Authorized Subsystems

A third method in which audit records are generated is through authorized subsystems such as the **lp**, **cron**, **terminal**, and **mem** subsystems. Sometimes a subsystem encounters inconsistencies or problems that make the writing of an informative audit record desirable. However, subsystems do not possess the **writeaudit** authorization and cannot directly write audit records to the subsystem.

Instead, the subsystems format the records just as a trusted application would, and present the records to the audit daemon process through a secure interface. The audit daemon, which is a trusted application, performs the task of writing the audit record to the audit device. This allows concise and informative audit records to be generated by protected subsystem processes without having to distribute the **writeaudit** authorization to these systems.

Accountability for Audit

The audit subsystem audits security-related system events and associates the events with a specific user. Users log into the system through the **login** program. This program performs authentication on the user to determine whether access is permitted. The login procedure has been enhanced to provide audit support for both successful and unsuccessful login attempts. When a user is successfully logged in, **login** stamps the user process with an immutable ID value called the login user ID (LUID). Regardless of the number of **setuid**(S) and **setgid**(S) system calls made by that process, the LUID does not change. Strict accountability is

maintained for the process and the user. A user process may still perform **setuid** and **setgid** system calls which are also audited. The audit records indicate the LUID of the process along with the effective and real user and group IDs of the process.

Audit Event Types

Every audit record, regardless of the originator, is stamped with an event type. For user process system calls, the event type is determined by the kernel audit mechanism, based on the system call and its outcome as previously discussed. For application or subsystem auditing, the process writing the audit record sets the event type. This event type is not changed by the audit device or by the audit daemon.

Event types are important because they classify the security event on the system. Both audit-record generation and reduction can be controlled based on event types. For example, if you are only concerned with users logging onto and off of the system, you can specify that event type for collection or reduction.

The audit subsystem provides a wide range of event types that strike a balance between granularity and relevant security classes. The event types supported are:

- System startup and shutdown
- Login and log out
- Process creation and termination
- Map objects (such as files) to subjects (such as processes)
- Make object available
- Make object unavailable
- Object creation
- Object modification
- Object deletion
- Inter-process communication
- Discretionary access changes
- Discretionary access denials
- Insufficient authorization
- Resource denials

- System administrator/operator actions
- Authorized subsystem actions
- Secure database actions
- Audit subsystem events
- Use of authorization.

You can selectively collect and reduce audit data based on these event types. The audit subsystem interface lets you build a list of event types for either the audit subsystem or the data-reduction program.

The subsystem uses event types to determine whether an audit record should be written to the audit trail. As the Audit Administrator, you have full control over what events get audited.

To control event type auditing, the subsystem contains a global *system audit event mask*, as explained below. The audit subsystem also maintains a mask of event types for each process on the system (explained in a forthcoming section). Both masks are bit representations of the integer event types to be audited.

Mandatory Auditing

To accurately maintain all the required information about a user process for meaningful audit output, the kernel audit mechanism always audits certain system calls. When auditing is enabled, this means that some events are audited even if no events have been selected by the audit administrator. We call these *mandatory* system calls. They are essential to the maintenance of the process state. For example, the **open**(S) system call may specify a relative pathname such as *../newfile*. The full pathname where the file is located depends on the current directory of the process which is set using the **chdir**(S) system call. The audit record containing the pathname *../newfile* could not be meaningfully reduced without prior knowledge of the value of the current directory.

The problem applies to the **close**(S) system call as well. This system call requires only a file descriptor as the argument to close a previously opened file. The **close** audit record would be insignificant unless the name of the object being closed is output in the record. However, unless the pathname is retained when the file is opened, there is no way to provide the pathname for the close.

Table 5.5 lists the audit event types affected.

Table 5.5.

Mandatory Audit Events

Event type	Always audited	Optionally audited
Make object available	**open, pipe**	**mount, opensem**
Object creation	**creat**	**link, mkdir, mknod, creatsem, sdget**
Map object to subject	**dup, exec, exece**	**fstatfs, getdents, stat, statfs**
Object modification	**execseg, unexecseg**	**chsize, stime**
Make object unavailable	**close**	**sdfree, umount**
Process create/delete	**exit, fork**	-
Process modification	**chdir, chroot, proctl, security, setgid, setpgrp, setuid**	-

Mandatory auditing is not limited to just the group of system calls listed in Table 5.5. The login event is the only mandatory trusted application audit record defined. When a user logs in, the login record contains an indicator of the terminal on which the login occurs. If that same user is logged into multiple terminals on the system, the actions of that user can be traced to a specific terminal.

System Audit Event Mask

The system event mask is global to the audit subsystem. You set this up using **sysadmsh** and can change it while auditing if you want to select a different set of events. The system event mask contains one bit for each event type; the bit is set to one when auditing is desired. This provides a fast test (using a bitwise operation) to determine if a newly-created record is enabled for auditing. The audit subsystem uses the system event mask to compute user masks when a new process is created through a login.

User-specific and Process Event Masks

You can override the system-wide event mask for any user by setting up a *user-specific event mask* in the user's protected password entry. Each process on the system has a *process event mask* which tells the system what to audit for that process. When a user logs in, the **login** program looks up the user-specific event mask and sets the process event mask for the login shell as follows.

The user-specific event mask has one of three values for each audit event type:

- Always audit this event
- Never audit this event
- Use the system audit event mask

For each audit event type, the process audit mask is set from the user-specific mask if it indicates that the event is always or never audited. Otherwise, the process audit mask is set from the system audit event mask. In most cases, the user-specific event mask will be set to the third value for all audit events, which will cause the system default to apply to that user. You can use the user-specific mask to audit either more or less information about users that you trust either less or more than the rest of the user population.

Effective System Auditing

You should follow certain guidelines to use the audit subsystem. The subsystem is designed to offer flexible performance and reliability and let you collect the audit data that you want. Audit-record generation supports *pre-selection* of audit events, user IDs, and group IDs. Pre-selection is valuable if you want to concentrate on a specific user or group of users for some reason (when particular users have a pattern of attempting access to files to which they are not permitted). Event types may also be used for pre-selection such as auditing only log-in and logoff events. Pre-selection also provides disk space savings benefits because the amount of audit records written to the collection files by the subsystem is reduced. There is however a drawback to using pre-selection. If a system security violation occurred and that event or the user that perpetrated the event was not selected for audit, the record of the action is lost.

For this reason, it is more conservative **not** to pre-select the audit events and users/groups, but instead to perform full auditing. The benefit is that any security-related event that occurs is recorded in the audit trail. The disadvantages of full auditing are that it consumes a lot of disk space and adds overhead to the system.

You can then combine full auditing with *post-selection* to examine only records of interest. Post-selection provides for the selective examination of the audit trail based on event types, user IDs, group IDs, and object names, as well as date and time of record generation. In all, the audit subsystem combined with the data reduction/analysis utility provides you with the flexibility to trade between system performance and disk capacity with pre-selection, and the convenience of full auditing combined with post-selection.

Administrative Concerns

The administration of the audit subsystem is the key to effective auditing. Through careful setup and use of the audit subsystem, you have a powerful tool that helps keep the system secure and identify problems when they do arise. The subsystem is designed to be very complete in terms of audit event coverage both from kernel actions and from the use of system utilities. It is also designed for reliability and to minimize the impact on the performance of the system as a whole.

How well the subsystem meets your goals depends on proper administration of the system. You control the tradeoff between reliability and performance using audit parameters. Improper setup can result in poor performance, loss of audit data, or both. For example, setting the audit event mask to govern event types audited by the subsystem is critical. For instance, if event pre-selection does not include login events, a penetration of the system through a dial-up line might go undetected. Therefore, it is vital that you carefully consider the following three items:

- Performance goals
- Reliability goals
- Audit trail requirements

Performance Goals

When estimating the impact of the audit subsystem on the performance of the system, it is important to consider the actions that must be performed by the subsystem. The audit subsystem device driver is the focal point for the collection of audit records from all sources and is responsible for writing those records to the audit trail. The driver writes to a collection file that is shared by all processes being audited in the system. This situation is similar to an airline reservation system where multiple clerks are accessing a common database. Lockout mechanisms must exist to prevent the intermixing of audit records and to insure the consistency of the database. The same is true of the audit subsystem collection files.

An internal buffering mechanism and a write-behind strategy tries to minimize the impact of multiple, simultaneous writers to the collection file. This lets the subsystem service audit records from processes and applications while collection files are being written in parallel. You can tune this mechanism for how much buffering is used and how frequently data is written to the collection file.

Reliability Goals

Equally important to the system's performance is the reliability of the audit trail produced. Traditional UNIX systems lack the element of preserving filesystem integrity when a system crash occurs. This stems from the fact that I/O is accomplished using a pool of buffers that are (mostly) written asynchronously. Thus, changes made to files may not actually be recorded on disk at the time of a system crash.

This is unfortunate since the events leading up to a system crash are the ones that are most interesting to you from an audit standpoint. It is highly desirable to minimize any potential data loss from the audit subsystem as the result of a system crash. To meet this objective, the audit subsystem uses a facility called synchronous I/O that causes audit collection buffers and collection file inodes to be updated immediately as they change. This minimizes the potential amount of data lost as the result of a system crash.

There is a direct correlation between the degree of data reliability and the performance of the audit subsystem. Audit records that are generated by the kernel audit mechanism, trusted applications, and protected subsystems are typically 40-60 bytes in length. If each record is written to the disk synchronously as it is presented to the subsystem, the result is poor performance; the I/O system gets flooded because of the high rate at which these records are generated. The solution is to buffer the records and write them together to the audit trail at selective intervals. These intervals may be determined by elapsed time or an accumulated data threshold. Again, the choice is yours.

Audit Trail Requirements

The final area critical to audit subsystem administration is determining what needs to be audited. Pre-selection options for record generation can be used to fine tune the audit trail to concentrate on an event or several events. For instance, the system may be limited in use to a small group of people but is left unattended at night. Additionally, several dial-in lines are provided for after-hours work. You may only be concerned with accounting for who uses the system and when. In this case, pre-selection may be used only to audit login and logoff events. Attempts to penetrate the system by unauthorized users would then be audited as unsuccessful login attempts.

Audit may also be focused on specific users or groups of users. This lets you concentrate on suspected violators of security policies. The less auditing requested, the less impact the subsystem has on the system performance. Full auditing creates an extensive and detailed record of system events, but also requires the most resources to accomplish. However, it is often better to have recorded the events and to use the reduction tools to discard unwanted records later than not to have the records that are really needed to examine a problem. This decision depends on the degree of security you wish to impose.

It is important to understand the definition of an audit session with respect to the subsystem. A session is intended to correspond to an interval from the time the system is booted until the system is taken down. To reduce the amount of data written to the audit trail, the goal of the subsystem was designed to minimize the size of each audit record. Consequently, the state of a process is defined by a sequence of audit records rather than being indicated completely in each record. The space and time savings of this approach are tremendous but require that careful administration be used to avoid pitfalls.

If the audit subsystem is disabled while the system is running and later re-enabled, a new session is created. A session is defined as the sequence of collection and compaction files containing the audit records associated with a specific time interval. This may result in two (possibly more) sets of audit files for a single system lifetime. Some processes that are audited in the second or subsequent session might have been created during the first session. Consequently, a session may not contain all of the relevant process state needed for a certain process. In turn, this can lead to incomplete record reduction. This applies mostly to file names and typically only in the case of relative (rather than absolute) file names. You can avoid this problem by disabling auditing only by taking the system down.

Audit Procedures

This section addresses how to set up, initiate, modify, and terminate the auditing of the system. Each of these functions is accessed through the **sysadmsh**. The top level of the audit functions are reached with the **System→Audit** selection, and are as follows:

Enable/Disable Audit initiation and termination.

Collection Audit Criteria Selection

Configure the subsystem and control the events and users that are audited. In addition, control can be exercised over setup parameters that affect the performance and reliability of the subsystem and its data.

Report Audit Data Reduction/Reporting

Create and save selective reduction files, reduce audit sessions using selection files, and remove reduced files after they are no longer needed.

Files Audit Record File Maintenance

Examine what audit sessions have accumulated on the system, archive audit sessions to backup media, restore previously saved audit sessions, delete unwanted audit sessions, and start and stop auditing as needed.

The audit procedure consists of three stages that are described in the sections that follow:

1. Setting up data collection.
2. Enabling auditing.
3. Generating audit reports.

Setting Up Data Collection

To specify the data to be collected and where it is to be stored, return to the desktop and double-click on the Sysadmsh icon to bring up the main **sysadmsh**(ADM) menu. Then make the following selection:

System→Audit→Collection

The following selections are displayed:

Directories	Display or modify audit collection/compaction file directory list
Events	Display or modify system audit collection type mask
IDs	Display or modify list of users and groups audited
Parameters	Audit subsystem parameters
Reset	Change collection rules back to the default values
Statistics	Display statistics of current audit session.

Select the information you wish to supply; each selection is discussed in the sections that follow. The collection information that you select is stored in a parameter file. The system is distributed with a default parameter file, but you should modify this file to meet your needs. Once initiated, the subsystem audits events as directed by the parameter file until the parameters are modified, auditing is terminated, or the system is halted. Note that some parameters may be modified while auditing is in progress; others are valid only at the time audit is initiated. As each area of configuration is described, static and dynamic values are pointed out.

Audit Directories

Both collection files, generated by the subsystem, and compaction files, generated by the audit daemon, are written to directories you specify. An audit session may contain files written to many different directories. At the conclusion of a session, only the compaction files remain because the collection files are removed by the subsystem as they have been read by the audit daemon. You do not need to keep track of the directories into which files are written since a session log file maintains this information.

You can improve the system's performance by placing the audit directories on a filesystem that resides on a different physical device from the rest of the filesystems. This reduces contention for disk resources. Also, audit requires large amounts of space, even with compaction. The subsystem warns you when disk space is low, and it eventually disables auditing if the free space of a filesystem is too low. For this reason, multiple directories are supported by the subsystem and the daemon. If an error occurs in writing to a directory or if space is exhausted, the subsystem and the daemon attempt to use alternate directories to continue.

Enter each file name as an absolute path name. There is no artificial limit on the number of directories you may specify. If no directories are specified, the subsystem and the daemon create all files in the root filesystem using the reserved audit subsystem directory */tcb/auditttmp*, the default configuration file setup.

Audit Event Mask

As discussed earlier under "Audit Event Types," there are a number of audit events that can be selected; these are shown in Table 5.6.

Table 5.6.

Audit Events

A. Startup/Shutdown	B. Login/Logoff
C. Process Create/Delete	D. Make Object Available
E. Map Object to Subject	F. Object Modification
G. Make Object Unavailable	H. Object Creation
I. Object Deletion	J. DAC Changes
K. DAC Denials	L. Admin/Operator Actions
M. Insufficient Authorization	N. Resource Denials
O. IPC Functions	P. Process Modifications
Q. Audit Subsystem Events	R. Database Events
S. Subsystem Events	T. Use of Authorization

Each event type is displayed and corresponds to a letter at the top of the screen. For those events that are to be audited, the event type should be specified with a "Y". Those event types that are not to be audited are excluded using the "N" option. Use **Space** to toggle an entry from "Y" to "N" and vice versa. Use the arrow keys to move from entry to entry. This event mask can be modified and dynamically altered for the current audit session and/or may be written to the parameter file to take effect on future audit sessions.

User and Group Selection

The User and Group fields can be used to dynamically alter the audit selection for the current session or may be used to affect the next session. Selection of users and groups may be done many times within the same session. If no users and groups are selected, the effect is to eliminate all users and groups from specific selection. This means that all processes on the system are subject to auditing.

Audit Subsystem Parameters

You can alter some audit parameters to tailor auditing to the needs of a system. Some of these parameters relate to the earlier discussion on performance and reliability tradeoffs. This should become more apparent now. The parameters are as follows:

Write to disk...

These two parameters control the frequency with which audit data is written synchronously to the audit collection file from the internal audit buffers. Flushing may be controlled either by the amount of data that accumulates before writing or after a specific time interval. The latter is valuable when small amounts of data are generated and the frequency of the record generation is spread out over time. You can specify both byte count and time lapse flushing. The time interval is always specified in seconds.

Performance may be adversely affected through a poor choice of either value. Writing too frequently slows the system becomes with excessive I/O traffic. On the other hand, when these values are too large, the potential for data loss increases if the system crashes. A good rule of thumb is to flush each time a single internal buffer fills. Thus, setting the flush-byte count to 1024 (the size of an internal buffer) is usually sufficient.

Wake up daemon...

This parameter controls the audit daemon. This daemon continually reads the audit device and retrieves records written to the collection files. These records are then compacted and written to compaction files that can later be reduced. To maximize the effectiveness of the compaction algorithm, the daemon needs to read blocks of data between 4 and 5 Kbytes. This requires special handling by the subsystem as a typical process read returns when any data is available rather than waiting for a specified amount of data to accumulate. For maximum effectiveness, this parameter should range from 4 Kbytes to 5 Kbytes. The default value is 4 Kbytes.

Collection buffers

This lets you specify the number of collection buffers for the subsystem to use. It uses these internal collection buffers to gather audit data for writing to the collection file. Multiple buffers are used to increase the efficiency of the system since all processes essentially share the buffer space attempting to write records. By providing multiple buffers, processes can deposit records and continue execution without blocking even if an I/O is occurring on previous buffers. A minimum of two buffers is required. Most systems cannot effectively use more than 4-6 buffers to avoid performance problems. There is no deterministic way to calculate the optimum number of buffers. Generally, base this value on the expected process load of the system.

Collection/Audit output file switch...

These two parameters let you specify the maximum size that collection and compaction files may grow before a new file is created. Choosing a small value for either parameter results in excessive file switches. Since compaction files are permanent, this can also lead to a proliferation of small files on the system. Choosing values that are too large creates a situation where audit collection files use large amounts of disk space even though they have been partially read by the audit daemon and could otherwise be discarded. The size of audit compaction files can be controlled because these files remain on the system until reduced and removed. It is desirable that these files be of reasonable size to work with, including being able to save and restore them easily. The default value for the collection files is 50K bytes, and the compaction files are 1 megabyte. Make sure that the maximum size chosen for the compaction files does not exceed the **ulimit** established for the system which governs the largest size a user file may be.

Compacted output files

This option is provided should non-compacted audit files be desired. There is no compelling reason why this option should be exercised because compaction does not require large amounts of additional processing time and the resultant disk savings are typically greater than 60 percent. The compaction algorithm is contained in the audit daemon user process, not performed in the kernel portion of the subsystem.

Enable audit on system startup

If yes, this starts auditing automatically each time the system is rebooted. This field is only displayed with the View option; it is set according to whether auditing has been enabled or disabled. If auditing has been disabled, then auditing will be disabled at startup.

Shut down auditing on disk full

This option allows the system to automatically shut down if the system runs out of disk space, thus avoiding data corruption.

Change parameters for this/future session

The last two options on the screen let you dynamically alter the current session and/or make the changes a permanent part of the audit parameter file for future sessions.

Summary of Current Statistics

A final option provided by the Collection menu is the retrieval of the current audit session statistics. This provides information on the current session number, the number of collection and compaction files, the number of records written by the kernel audit mechanism and the number written by applications, as well as other information. If auditing is not currently in effect, no statistics are displayed.

Figure 5-1 is an example of the *Summary* selection.


```
        *** Audit Subsystem Statistics ***

Current Audit Session-6
Current Collection File Sequence Number-1488

Total count of audit data written:       7659433
Total count of audit records written:    156666
Audit records written by applications:   81
Audit records written by system calls:   155083
System calls not selected for audit:     751889
Total number of audit device reads:      2977
Total number of audit device writes:     324
Total number of collection files:        1489
```

Figure 5-1. Audit Collection Summary Example

Enabling/Disabling Auditing

To switch auditing on or off, return to the desktop and double-click on the Sysadmsh icon to bring up the main **sysadmsh**(ADM) menu. Then make the following selection:

System→Audit→Enable
System→Audit→Disable

The enable function uses the current audit parameter file to perform the subsystem initialization. The disable function is available from the same menu and causes a graceful exit from auditing (at which point all collection files have been read by the daemon and compacted). The daemon then terminates leaving only an audit session log file and the session compaction files.

Remember that disabling audit and then re-enabling without a system reboot may cause the loss of some process data required to maintain the process state. If the reason for stopping audit is to modify certain parameters, note that most subsystem parameters can be modified while audit is running. Both enable and disable functions have confirmation screens that must

be acknowledged before the function is completed by **sysadmsh**. When auditing is enabled or disabled, a message is displayed indicating the status of auditing at reboot time; if disabled, auditing will be disabled at system startup and if enabled, auditing will be enabled again at startup.

Maintaining Audit Files

The audit File Maintenance functions are accessed under the following **sysadmsh** selection:

System→Audit→Files

The following File functions are available:

List List audit session files on system.

Backup Back up an audit file session to backup media.

Delete Remove an audit session file.

Restore Restore an audit file session from backup media.

An audit session consists of a session log file and a group of compaction files generated between an enable and disable of the audit subsystem. Each collection file and compaction file created during a session is uniquely numbered with the session in which it was created. When sessions are completed, only the log file and the compaction files remain. The File Maintenance functions examine what sessions are still on the system and let you remove sessions no longer wanted.

Listing Audit Records

This selection is straightforward; it lists the files available in the audit directories.

Backing Up Audit Records

Since audit sessions require a large amount of disk space, it is often necessary to archive audit data and either reduce it later or retain it for some period of time in case it is needed to analyze problems that are not immediately detected. The backup/restore interface provides this capability. The *Backup* option requires a session number as input. This can be obtained by generating an audit report (see below). After selecting backup, you must select an output device for the backup. This can be any removable media available on the system.

NOTE: Auditing is a great consumer of disk space. Depending on how many users on your system and how many events are audited, it may be necessary to back up and remove session files on a weekly basis. If you have scheduled backups, it will probably not be necessary to use the audit backup selection. Again, it is important to remove the files to free disk space after they are backed-up.

Similarly, sessions that have been backed up onto removable media using the interface program may be reloaded using the *Restore* option. To do so, insert the media containing the saved session files into the restore device, and specify the device name.

Removing Files

The *Delete* selection is provided for the removal of audit sessions. Sessions may be archived to backup media and removed to make room on the filesystem for more audit files. Sessions are removed using the session number. A typical scenario would include generating a report (see below) to determine what sessions exist and which of those could be removed. The session number is then presented to the *Delete* option to delete all of the files associated with that session.

Generating Audit Reports

The reduction program uses a file called a *selection file* to perform post-selection of audit records. This file is built by the Audit Administrator interface program based on your input. You can build and save multiple files, each with a different set of selection criteria. Reduction may then be run several times on the same session data with a different selection file each time. Thus, you can build and save selection files used frequently in data reduction. When the actual data reduction is needed, you can use the files already built.

To examine the audit trail of any session, select the following from **sysadmsh**:

System→Audit→Report

The following options are available:

List List all selection files available.

View View the parameters stored in a selection file.

Create Create a new selection file.

Modify Modify an existing selection file.

Delete Delete an existing selection file.

Generate Make a reduction run, specifying audit session and selection file.

As discussed earlier, audit collection criteria represents the first level of audit selection. After the data is gathered, it can be further processed, or *reduced*, to generate a useful collection of data about a specific aspect of system operation. The data reduction menus let you select to reduce and determine what records are desired. The *Generate* option supports a wide range of post-selection criteria that helps you target specific events, users, or objects.

The last option displays the reduced output from an audit session. This requires the session number and the selection file, which may be any of the selection files built using the selection file create or update options.

The options for *List*, *View*, *Create*, *Modify*, and *Delete* are used for selection-file maintenance. The actual contents of the selection is discussed more fully in a later section. The **System→Audit→Report** selection is used to invoke the next level screen to perform the required selection file function. As is indicated by the option names, selection files may be created, updated, or deleted as necessary.

The following criteria can be selected to reduce audit records:

Event types — Each event type desired is marked with a "Y". Event types to be excluded are left blank or filled with an "N". Events not selected cause those records to be discarded from the output.

Start and Stop times — If a security-related event was suspected between certain times of the day, you could use this feature to select those records that were generated during that time period. This could serve to concentrate the analysis on those records that are likely to reveal what has happened.

Users/Groups — Both users and groups of users may be singled out for audit. If a certain user account was the target of a penetration, you could select only those records that were generated from user or group IDs that matched that user. This permits the record search to be concentrated on suspected accounts.

Files — Files (object names) can also be used to select audit records from the output. For records that contain multiple object names, if a specified name matches *any* object in the record, the record is selected. The object names must be specified as absolute path names because all object names are resolved from relative to absolute names by the reduction program.

Any combination of the above criteria can be used. For instance, selectivity on time interval, user ID, and object name may be combined for a single session. If a record is within the specified time interval, was generated by a selected user, and has one of the selected objects in the record, then it is selected for output.

There is a precedence for record selection that governs the combination of the selection criteria. If the audit event type is not specified, the record is not selected, regardless of other criteria. Likewise, if time stamp selection is enabled and the record does not meet the criteria, the record is not selected. If the record passes the selection criteria for event type and time, then the record is selected if it has a user ID (login, effective, or real), group ID (effective or real), or an object in the record that is specified in the selection file. If no users, groups, and objects are specified, only event type and time selection is performed.

Understanding Data Reduction

The *reduce* option of the Audit Report menu is as important as all of the components responsible for generating audit records. This utility converts the compacted audit trail data into an organized and readable collection of records that help you to isolate system problems. The reduction features allow you to reduce the time required to examine records of interest. You can concentrate on a specific user, groups of users, object names, event types, and records generated in a certain time interval. These can be used together to provide a powerful selection mechanism.

To interpret the audit trail, you need to understand the records produced by the program and what they mean. Remember that audit records come from three sources: system calls, trusted applications, and protected subsystems. Record formats differ greatly among these three sources. Further, system calls differ greatly from one another in content because of the specific function being performed. For instance, a process creation, **fork**(S), need only indicate the process ID of the newly-created process and the ID of its spawning process (parent). However, for an **open**(S) system call, an object is being acted upon and the name of that object must be recorded. For system calls like **mount**(S) and **link**(S), still more information must be recorded; each requires that two object names be recorded. The reduction utility sorts records presented to it and outputs the information in an organized manner.

Output records can be classified into two types: system call records produced by the kernel audit mechanism and application audit records. Some items are considered common to all output records. For instance, the date and time of the record and the process ID associated with the record are printed for each type. Beyond this, the content of a record depends on what was audited.

System Call Record Formats

System call records account for the majority of the records in the audit trail. The operating system contains over 60 system calls. Not all of these system calls are audited as only some of these are deemed to be security-related. Slightly over half of the system calls have the potential to create an audit record. Some system calls support multiple functions (such as **fcntl**(S), **msgsys**(S), **shmsys**(S), and **semsys**(S)) that may only generate audit records for certain functions. For instance, the **fcntl**(S) system call allows files to be opened by duplicating open file descriptors and also permits file specific flags maintained by the kernel to be retrieved. The first case constitutes an auditable event, making an object available to a subject, while the second has no real security relevance. Furthermore, system calls may perform functions that are considered auditable events but are not enabled by the system event mask at the time.

For the most part, the output of system call records is the same for all calls. Variations exist because some system calls operate on objects (such as **open**(S)) and the object name is contained in the record. Each contains at least the time, date, process ID, system call name, event type, login user ID, real user and group IDs, effective user and group IDs, and an indicator of success or failure for the call.

Each output record contains these basic information fields and others depending on the system call. The basic record is shown in Figure 5-2. This illustrates the common header along with the system call and result fields.

```
Process ID: 68          Date/Time: Sat Mar  5 13:25:09 1988
Luid: root  Euid: root  Ruid: root  Egid: root  Rgid: root
Event type:
System call:
Result:
```

Figure 5-2. Common Output Record Header.

Each system call is classified into a system event type based on the actions that are performed. This is used to describe the event type of the system call. The actual system call name is given. In most cases this uniquely identifies the action. Unfortunately, some UNIX system calls are overloaded, causing a system call entry point to be used to accomplish multiple actions. For example, **msgsys**() is the system call entry for message queue IPC operations. This single entry point is used to call **msgget**(S), **msgop**(S), and **msgctl**(S) to perform certain IPC functions.

System calls like this are not self-explanatory. The audit subsystem is aware of these overloaded calls and provides additional information to identify the particular function. For system calls that succeed, the result is specified as successful. For each that returns an error, the error is used to provide additional record classification. For instance, an **open**(S) that fails from lack of permission is classified as an access denial. An unsuccessful system call that generates an audit record indicates the error in the result field.

The system call output records can be divided into two groups. The first group contains records that do not require pathnames in the audit record. For instance, the **fork**(S) system call is audited to track new processes as they are spawned into the system, but the audit record does not require a pathname. On the other hand, **open**(S), returns a file descriptor for the specified pathname. Subsequent operations, like **close**(S), use the file descriptor. To provide meaningful audit records, this second type of record must contain the pathname. Using the reduction program, this pathname is associated with all further actions on that file, even though the action may have been performed with a file descriptor.

Figure 5-3 lists audited system calls that do not contain pathname information.

pipe	fork	kill
setuid	setgid	exit
read	setpgrp	msg
sem	shm	write

Figure 5-3. System Calls without Pathnames.

An output record from one of the above system calls uses the generic record mask described in Figure 5-4. The example shown in the following figure illustrates the output record from a successful **setuid**(S) system call.

```
Process ID: 6381        Date/Time: Tue Mar 15 11:25:19 1988
Luid: blf  Euid: blf  Ruid: root  Egid: root  Rgid: root
Event type: Modify process
System call: Setuid
Result: Successful
```

Figure 5-4. Setuid(S) System Call Record.

Similarly, the following figure shows the output record from a **setuid**(S) system call that failed due to a lack of permission on the file. Notice that the event type classification is different and that the error is reflected in the result field.

```
Process ID: 6381        Date/Time: Tue Mar 15 11:25:19 1988
Luid: blf  Euid: blf  Ruid: blf  Egid: guru  Rgid: guru
Event type: Modify process
System call: Setuid
Result: Failed (EPERM)-Not owner
```

Figure 5-5. Access Denial Output Record.

Many system calls in this group generate additional information in the output record to help clarify the audit trail. The semaphore, shared memory, message queue and **security**(S) system calls are overloaded. They map to multiple functions. These audit records identify the specific function being performed and also the affected object (for example, shared mem-

ory). **close**(S), **dup**(S) and **fcntl**(S) operate on file descriptors that were mapped from pathnames. An output record indicating a **dup**(S) of a file descriptor would not be very useful since it does not uniquely identify the file. Thus, *reduce* correlates the file descriptor to a pathname and prints the pathname in the record.

Even though the **read**(S) and **write**(S) system calls are listed in the Figure 5-3, they are audited only in certain circumstances and neither has a dedicated output record. Both system calls are audited only for the first occurrence for a file. Subsequent reads and writes do not need to be audited as they provide no additional information. The audit records are used by *reduce* to track the state of the file. When the file is closed due to **exec**(S), **exece**(S), **close**(S), or **exit**(S), the name of the object and an indicator of whether the file was read or written is included in the system call record for the action that caused the file to be closed. This is illustrated in Figure 5-6.

```
Process ID: 421         Date/Time: Sat Mar  5 17:15:09 1988
Luid: blf  Euid: blf  Ruid: blf  Egid: guru  Rgid: guru
Event type: Make object unavailable
System call: Close
File Access-Read: Yes  Written: No
Object: /tmp/datafile
Result: Successful
```

Figure 5-6. Close(S) System Call Record.

The second group of system calls, shown in Figure 5-7, contains pathnames as part of the output record. The pathname represents the target of the system call. Two of the system call records actually contain two pathnames: **link**(S) and **mount**(S).

open	unlink	creat
exec	chdir	mknod
chown	chmod	stat
umount	exece	chroot
link	mount	

Figure 5-7. System Calls with Pathnames.

Each of the system calls in Figure 5-7 takes one or more pathnames as arguments to the call. The pathnames are audited and become an important part of the reduction process. Output records for these calls indicate the object name acted upon. This name is also retained by the reduction program and where applicable is associated with the file descriptor returned by the system call. This provides a mapping for other system calls like **dup**(S) that operate on the file but do not contain the pathname. Figure 5-8 shows an output record generated from a **creat**(S) system call. The record format is the basic format augmented by the pathname.

```
Process ID: 64          Date/Time: Sat Mar  5  23:25:09 1988
Luid: root  Euid: root  Ruid: root  Egid: root  Rgid: root
Event type: Object creation
System call: Creat
Object:  /tmp/daemon.out
Result: Successful
```

Figure 5-8. Output Record with Pathname.

All of the calls in this group use the same format for pathnames. Two calls, **link**(S) and **mount**(S), operate on two pathnames: a source and a target. Both names are audited and reflected in the output record by the reduction program. A typical record produced by a **link**(S) system call is shown in Figure 5-9.

```
Process ID: 14231       Date/Time: Thu Mar 16  03:25:39 1988
Luid: lp  Euid: lp  Ruid: lp  Egid: lp  Rgid: lp
Event type: Object creation
System call: Link
Source:  /tmp/printfile
Target:  /usr/spool/lp/lp3014
Result: Successful
```

Figure 5-9. Output Record with Two Pathnames.

Two other records in this group generate special output records. These are **chown**(S) and **chmod**(S), which are used to alter discretionary access permissions and file ownership for objects. Due to the security-relevant nature of their actions, the previous and new values of the object owner, group, and mode are output in the record. Figure 5-10 illustrates the output record from a **chmod**(S) system call.

```
Process ID: 6841        Date/Time: Sat Mar  5 13:25:09 1988
Luid: blf  Euid: blf  Ruid: blf  Egid: guru  Rgid: guru
Event type: Discretionary Access Change
System call: Chmod
Object:  /tmp/demo/newfile
Old values: Owner-blf  Group-guru  Mode-100600
New values: Owner-blf  Group-guru  Mode-100666
Result: Successful
```

Figure 5-10. chmod(S) System Call Record.

Application Audit Records

There are six different types of audit records generated by application programs. The formats for these are similar. Unlike system calls, any record produced in one of the six categories is always formatted identically, although the information varies. The categories are:

- Login and Logoff events
- Audit Subsystem events
- User Password events
- Authorized Subsystem events
- Protected database events
- Terminal and User Account Lock events

Each record contains some information common to all audit output records. This includes the process ID, the time and date, and the audit event type. The remainder of the output record depends on the record type. The record-specific fields are described in the following sections.

Login/Logoff Record

All attempts to log into the system are audited by the login program. This is true of successful as well as unsuccessful attempts. This creates an important trail of user accesses to the system and also a trail of attempted accesses. You can use the audit records for log-in/logoff to determine who actually used the system. It is also valuable in determining if repeated penetration attempts are being made. The operating system supports the option of locking terminals after a certain number of attempts and this event can also be audited. Thus, you have all tools necessary to monitor (and prevent) access to the system.

Each login record contains an indicator of the specific action that was audited. The three possibilities are: successful login, unsuccessful login, or logoff. All successful logins and logoffs result in an audit output record that indicates the user account and terminal of the log-in session. For unsuccessful attempts, the user name is meaningless, since the attempt failed. In this case, only the terminal on which the attempt occurred is output along with the basic record fields. Figure 5-11 illustrates the output from a successful login.

```
Process ID: 2812       Date/Time: Fri Mar  4 10:31:14 1988
Event type: Login/Logoff Activity
Action: Successful login
Username: blf
Terminal: /dev/tty2
```

Figure 5-11. Successful Login Audit Output Record.

User Password Record

All attempts, successful or not, to modify a user account password are carefully audited by the authorization subsystem. For security reasons, audit records for these events contain no password text, but only indicate the account and action that was audited. The actions are classified into successful password change, unsuccessful change, and lack of permission to change the password. Figure 5-12 shows an audit record for an unsuccessful password change.


```
Process ID: 7314       Date/Time: Tue Mar  1 18:30:44 1988
Event type: Authentication database activity
Action: Unsuccessful password change
Username: blf
```

Figure 5-12. Unsuccessful Password Change Audit Record.

Protected Database Record

Programs that maintain and modify the system's protected databases audit all access attempts and unusual circumstances associated with the databases. This may range from integrity problems to security-related failures. In addition to the record header and the specific audit action, the output includes the name of the program detecting the problem, the object affected by the problem, expected and actual values, and the action and result of the event.

```
Process ID: 7314       Date/Time: Tue Mar  1 18:30:44 1988
Event type: Authentication database activity
Command: authck
Object: Protected password database
Value: Expected-0 Actual-0
Security action: /tcb/files/auth/code
Result: extraneous file in protected password hierarchy
```

Figure 5-13. Protected Database Output Record.

Audit Subsystem Record

Events that affect the operation of the audit subsystem itself are audited very carefully. The **sysadmsh** audit selections and the audit daemon, **auditd**, both generate audit records for functions they support. Additionally, the audit device driver also writes audit records for certain function requests. The functions audited include the following:

- Subsystem initialization
- Subsystem termination
- Subsystem parameter modification
- Audit daemon enabled
- Audit daemon disabled
- Subsystem shutdown
- Subsystem error

Each output record includes the common header information along with an indicator of the function audited. This provides an accurate accounting of all attempts to affect the operation of the audit subsystem. Figure 5-14 shows an actual audit record written to indicate the startup and initialization of the subsystem.

```
Process ID: 517       Date/Time: Wed Mar  2  8:30:04 1988
Event type: Audit subsystem activity
Action: Audit enabled
```

Figure 5-14. Audit Subsystem Output Record

Protected Subsystem Record

Each protected subsystem can generate audit records through the audit daemon. These records indicate unusual conditions detected by the subsystem. For instance, if a subsystem encounters permission problems with a file or is denied service due to lack of memory or some other resource, the subsystem generates an error message to that effect. You can use these records to help maintain the security of the system.

Aside from the normal record header output, the subsystem records contain a command name, an action and a result. The command name is the command that detected the inconsistency and wrote the audit record. The action and result describe the action taken by the subsystem and the problem detected Figure 5-15 shows a subsystem-generated audit record.

```
Process ID: 2812       Date/Time: Fri Mar  4 10:31:14 1988
Event type: Authorized subsystem activity
Subsystem: System Administrator Subsystem
Security action: Update /etc/rc
Result: Cannot open for update
```

Figure 5-15. Authorized Subsystem Audit Output Record.

Terminal/User Account Record

User accounts or terminals may become locked if the number of unsuccessful login attempts, as stored in the Authorization database, is exceeded. For instance, if a terminal is used to enter the system and the result is a series of unsuccessful logins, the login program may lock the terminal after a specified number of tries. Similarly, if a user attempts to login to an account and fails repeatedly, that user account may be locked. Locking accounts and terminals prevents further access until you, the System Administrator, clear the lock. A terminal or user account lock may an attempted penetration of the system. These audit records contain the usual header information along with an identifier of the user account or terminal.


```
Process ID: 517       Date/Time: Wed Mar  2  8:30:04 1988
Event type: System administrator activity
Action: User account locked by system administrator
Username: root
```

Figure 5-16. User Account Lock Output Record.

Audit Problem Areas

The following situations present difficulties with respect to auditing.

Disk Space

The audit subsystem can generate a large number of audit records. Even though the records are fairly small, the storage required to maintain them can grow quite large. As a consequence, care must be exercised in administering the system. Auditing should be directed to disks that have a good deal of space available. The subsystem has built-in protection mechanisms that warn when the audit device is getting low on space. If the situation is not rectified and the amount of disk space remaining goes below a certain threshold, the subsystem attempts to switch to a new audit directory. For this reason, alternate audit directories should be placed on different filesystems. Whenever the subsystem encounters an I/O error, it attempts to audit to a new directory in the list.

System Crashes

Most systems crash at some time, despite every effort to provide a resilient base. If a system crash occurs, there is a potential for data loss in the audit trail due to buffered output records and inode inconsistencies. The audit subsystem makes every attempt to use synchronous I/O for critical operations like buffer, inode, and directory flushing. However, this does not guarantee that data always makes it to the disk. This is especially true if a disk failure causes the system crash.

It is not uncommon to find filesystem damage on audit trail files upon re-boot. You may have no choice but to remove the audit files to clear up the problem. This compromises the audit trail somewhat but should pose no problem for recovering the filesystem from whatever damage occurred.

Subsystem Messages

The audit subsystem is resilient. I/O errors are handled by the subsystem by attempting to switch collection or compaction to a new directory. The same is true of recovery in cases where filesystem free space gets too low. There are situations where the subsystem may be unable to continue. If the disk media is corrupted or there is no filesystem space remaining, the subsystem terminates and prints a message to that effect on the system console. Any abnormal termination condition results in a console message that should help you determine the problem. In the case of system problems in general, the symptoms are not generally limited to audit alone. One problem that can occur upon removal and subsequent re-creation of the audit parameter file relates to duplicate session-building. Each time auditing is enabled, a new session is created. The session is defined by the log file and all of the compacted files generated during the audit period. The files are uniquely stamped with the session number for easy identification and use by subsystem utilities that need access to the files; the utilities may deal with session numbers rather than file names. If sessions are allowed to remain on the machine and the parameter file is modified such that the subsystem session number is reset, the result may be an attempt to create an audit file using the same name as a previous session. If this occurs, the old session should be archived and removed using the interface program before auditing is re-enabled.

Audit Terminology

An *audit collection file* is a file written by the audit subsystem device driver containing the raw audit data from all audit sources on the system including system calls, trusted applications, and authorized subsystems.

An *audit compaction file* is a file written by the audit daemon containing buffers of data read from the audit device driver. The data may be in either a compacted or non-compacted format, depending upon options selected at the time the audit session was started.

The *audit daemon* is a daemon process started when the system makes the transition to multi-user state. It reads the audit subsystem device to retrieve audit records, compacts these records, and writes them to a permanent compaction file for later reduction. The daemon also acts as an interface program that permits non-protected subsystems to write audit records to the audit device.

An *audit session* is the period of time from audit enable until audit disable. During this time, the audit data is stored in compaction files written by the daemon. Each session is uniquely numbered and each file that is part of the session contains this unique ID in the file name. A master file is used for each session to collect session information and session file names for later reduction.

The *audit subsystem* consists of the components that provide the secure audit services. This includes the audit device driver, the kernel audit mechanism, the audit daemon, the Audit Administrator interface, and the audit reduction program.

An *audit trail* is the collection of audit data records from an audit session that can be reduced into a report of system activity.

audit reduction is the transformation of raw audit trail data into output records containing dates, user IDs, file names, and event types. The output record describes the audited event in a readable text form.

configaudit is the kernel authorization that allows the audit parameters to be set for all users of the system.

The *event control mask* is the user-specific mask maintained in the Protected Password database on a per-user basis. This mask controls whether the user event mask prevails over the system default event mask when auditing is enabled. Each bit set in the control mask causes the event disposition mask to take precedence.

The *event disposition mask* is the user-specific mask used in conjunction with the event control mask for user audit event control. If the user event control mask has a bit set on, the corresponding bit entry in the event disposition mask determines whether the event is always audited or never audited. This holds true regardless of the system default event mask value.

An *event type* is a classification for each audit record. Security-related events on the system are classified into certain types that can be used to control audit generation or reduction. Every system action, regardless of success or failure, can be classified into an event type. This event type then determines the disposition of the record.

An *object* is an entity acted upon by a subject (such as files, shared memory segments, semaphores, pipes, or message queues).

post-selection is the selective use of collected audit data. Post-selection involves collecting audit data for all events and users so the audit trail is as complete as possible. Any security-related event is in the audit trail compaction files at the end of a session.

pre-selection is used to selectively control audit record generation. This allows certain users and events to generate audit records while others are discarded. The result is a more compact audit trail with less detail than if full auditing was used.

selection files are generated through the administrative interface to control the selective reduction of audit sessions. Selective criteria control the user, object, and event selection for output records.

A *subject* is an active entity that performs actions on objects, such as a process on the system that accesses files.

suspendaudit is a kernel authorization that suspends auditing.

The *system audit mask* is the default system event mask used to determine what events are audited when a user process mask does not take precedence.

The *user audit mask* collectively refers to the event control and event disposition masks that, together with the system default mask, control the generation of audit records on a per-process basis.

writeaudit is a kernel authorization that allows specific information to be recorded by the audit trail.

Filesystem Protection Features

Your system includes important filesystem features that extend the protection present on other UNIX systems. These features greatly enhance the security of the system. One of them, SUID and SGID bit clearing upon file writes, is passive in that it requires no action by you, the System Administrator, for it to occur. The other features are active, meaning that you can choose whether or not to use them. These active features, discussed below, include the special use of the sticky bit on directories and the use of promains. This section also covers the importation of files from others systems.

SUID/SGID and Sticky Bit Clearing on Writes

The operating system guarantees that the SUID, the SGID, and the sticky bits are cleared on files that are written. The reason for the clearing is to prevent program replacement in a SUID/SGID program or one that is meant to be memory-resident.

In Figure 5-17, the bit clearing is demonstrated twice (user input is boldface).

```
$ id
uid=76(blf)  gid=11(guru)
$ ls -l myprogram
-rwsrwsrwt    1 root   bin     10240 Jan 11 22:45 myprogram
$ cat sneakyprog > myprogram
$ ls -l myprogram
-rwxrwxrwx    1 root   bin     10240 Mar 18 14:18 myprogram
$
$ ls -l anotherprog
-rws------    1 blf    guru   83706 Dec 15  1987 anotherprog
$ strip anotherprog
$ ls -l anotherprog
-rwx------    1 blf    guru   17500 Mar 18 14:19 anotherprog
$
```

Figure 5-17. Bit Clearing Example

The first case demonstrates that the bit clearing occurs when writing files owned by another user. The second demonstrates that the bit clearing is even done on files owned by the same user. You should be aware that the clearing happens when files are replaced. Adjust any installation scripts to reset the proper modes.

With this feature, you can place the sticky bit on user programs without fear that the user can switch programs in the same file. This is a case where the added security lets you perform a function that the system can track, rather than simply not allowing the feature.

The SUID, SGID, and sticky bits are not cleared on directories. The SUID and SGID bits have no meaning for directories, while the sticky bit has a meaning for directories that warrant its remaining there. This is described next.

The Sticky Bit and Directories

Another important enhancement involves the use of the sticky bit on directories. A directory with the sticky bit set means that only the file owner and the super user may remove files from that directory. Other users are denied the right to remove files. Only the super user can place the sticky bit on a directory. Unlike with files, the sticky bit on directories remains there until the directory owner or super user explicitly removes the directory or applies **chmod**(C) or **chmod**(S) to it. Note that the owner can remove the sticky bit, but cannot set it.

You can gain the most security from this feature by placing the sticky bit on all public directories. These directories are writable by any non-administrator. You should train users that the sticky bit, together with the default **umask** of 077, solves a big problem area of less secure systems. Together, both features prevent other users from altering or replacing any file you have in a public directory. The only information they can gain from the file is its name and attributes.

In Figure 5-18, you can see the power of such a scheme.

```
$ id
uid=76(blf) gid=11(guru)
$ ls -al /tmp
total 64
drwxrwxrwt   2 bin      bin       1088 Mar 18 21:10 .
dr-xr-xr-x  19 bin      bin        608 Mar 18 11:50 ..
-rw-------   1 blf      guru     19456 Mar 18 21:18 Ex16566
-rw-------   1 blf      guru     10240 Mar 18 21:18 Rx16566
-rwxr-xr-x   1 blf      guru     19587 Mar 17 19:41 mine
-rw-------   1 blf      guru       279 Mar 17 19:41 mytemp
-rw-rw-rw-   1 root     sys         35 Mar 16 12:27 openfile
-rw-------   1 root     root        32 Mar 10 10:26 protfile
$ rm /tmp/Ex16566
rm: /tmp/Ex16566 not removed.  Permission denied
$ rm /tmp/protfile
rm: /tmp/protfile not removed.  Permission denied
$ cat /tmp/openfile
      Ha! Ha!
You can't remove me.
```

(Continued on next page)

(continued)

```
$ rm /tmp/openfile
rm: /tmp/openfile not removed.  Permission denied
$ rm -f /tmp/openfile
$ rm /tmp/mine /tmp/mytemp
$ ls -l /tmp
drwxrwxrwt   2 bin      bin       1088 Mar 18 21:19 .
dr-xr-xr-x  19 bin      bin        608 Mar 18 11:50 ..
-rw-------   1 blf      guru     19456 Mar 18 21:18 Ex16566
-rw-------   1 blf      guru     10240 Mar 18 21:18 Rx16566
-rw-rw-rw-   1 root     sys         35 Mar 16 12:27 openfile
-rw-------   1 root     root        32 Mar 10 10:26 protfile
$ cp /dev/null /tmp/openfile
$ cat /tmp/openfile
$ cp /dev/null /tmp/protfile
cp: cannot create /tmp/protfile
$ ls -l /tmp
drwxrwxrwt   2 bin      bin       1088 Mar 18 21:19 .
dr-xr-xr-x  19 bin      bin        608 Mar 18 11:50 ..
-rw-------   1 blf      guru     19456 Mar 18 21:18 Ex16566
-rw-------   1 blf      guru     10240 Mar 18 21:18 Rx16566
-rw-rw-rw-   1 root     sys          0 Mar 18 21:19 openfile
-rw-------   1 root     root        32 Mar 10 10:26 protfile
$
```

Figure 5-18. Sticky Bit Example

The only files removed are those owned by user **blf**, since the user was **blf**. The user **blf** could not remove any other file, even the accessible file */tmp/openfile*. However, the mode setting of the file itself allowed *blf* to destroy the file contents; this is why the *umask* setting is important in protecting data. Conversely, the mode on */tmp/protfile*, together with the sticky bit on */tmp*, makes */tmp/protfile* impenetrable.

All public directories should have the sticky bit set. These include, but are not limited to:

- */tmp*
- */usr/tmp*
- */usr/spool/uucppublic*

If you are unsure, it is far better to set the sticky bit on a directory than to leave it off. You can set the sticky bit on a directory with the following command, where *directory* is the name of the directory:

chmod u+t *directory*

Promains

A *promain* is the term for UNIX *Pro*tected Do*main*, where the invoker of a SUID program obtains some protection from the program. While operating on a part of the file tree chosen by the invoker, the SUID program has its access checking validated against both the owner of the SUID program and against the invoker. The detailed description of the model, along with some examples of use, is provided in **promain**(M). Promains are also discussed in **auths**(C) and **setauths**(S).

Promains are a tool intended for investigating SUID programs and are not of general value. You need to be aware of promains when helping users debug a problem in their environment.

Importing Data

Files and filesystems brought into the system from elsewhere are a threat to the system if not handled properly. The remainder of this section discusses techniques to use when importing files to your system.

Files

Do not take for granted the permissions on a foreign file. Not only are the **/etc/passwd** and **/etc/group** files different on each system, but the policies on differing systems dictate setting different modes. These considerations are critical when the imported files are system files.

To minimize your intervention and clean up after importing files, train everyone on the system to use archive program options that do not reset ownerships. Some versions of **tar**(C) support the **-o** option, which does not reset the owner and group of files. The files are owned by the user importing the files. The **cpio**(C) program only changes the ownerships of files when the invoking user is the super user. The archive programs generally reset the file modes to that described on the media containing the archive. In addition to having a mode that is more permissive than necessary, files can have the SUID, SGID, or sticky bits set. All of these situations can create security problems for you.

To minimize the effects of the archive permissions, use archive options that examine the contents without extracting anything. For example, the **-tv** option to **tar** and the **-tv** option to **cpio** let you see the modes of the files on tape and prepare for any ill effects when extracting files. When bringing in unfamiliar archives, first import files into a hierarchy not accessible to others. Then manually move the files, after adjusting the ownership and modes according to your system policy.

Filesystems

Mounting filesystems that were created or handled elsewhere have all the concerns above for importing files. Filesystems also bring with them two extra concerns. The first is that the filesystem may be corrupted. The second is that file permissions on the filesystem may not be acceptable for your system.

A filesystem brought from elsewhere may be corrupted. The data may be bad or it may be intended for another type of system. In either case, mounting a bad filesystem can cause the system to crash, the data on the imported file system to be further corrupted, or for other filesystems to go bad from side-effects. This is why the **mount**(ADM) command is reserved for the super user. The **fsck**(ADM) program should be run on all filesystems **before** they are mounted. If the filesystem contains system data, the **fixperm**(ADM) utility should also be run.

Imported filesystems can contain file permissions not suitable for your system. The super user of the imported filesystem may have set ownerships, sticky bits, special files, SUID/SGID bits, and file tree compositions incompatible with your system policies. Special files may exist with different ownerships and modes that you cannot allow. Programs with the SUID, SGID, or the sticky bit set elsewhere, when mounted, also work that way on your system and can create security problems.

You can use the **-s** option to **ncheck**(ADM) to locate some of the problem files before mounting. Filesystems, like files, should be scanned before they are mounted. The first time a filesystem is mounted in your control, it is best to mount it in a private directory so you may scan the filesystem manually before mounting it in its normal place. Examine the file organization, the owners and modes of the files, and the expected use of the filesystem.

Data Encryption

Additional protection is available in the form of data encryption software, the **crypt**(C) command. These features are described in *Using ODT-OS* in the *User's Guide*.

NOTE: The data encryption software is not included in your distribution, but is available by request only within the United States. You can request this software from your dealer or distributor.

Setting Directory GID Bit

By default, the GID (group identifier) of a newly-created file is set to the GID of the creating process/user. This behavior can be changed by setting the GID bit on a directory. Setting the GID on a directory results in a new file having the GID of that directory. To set the GID bit on a directory, enter the following command, where *directory* is the directory name:

chmod g+s *directory*

Setting Filename Truncation

By default, attempts to create filenames longer than 14 characters result in the error message "Filename too long." This can be changed to have long filenames silently truncated to 14 characters. The default behavior is mandated by POSIX FIPS requirements and is controlled by the ETRUNC kernel parameter. This parameter can be changed by invoking the **sysadmsh** selection **System→Configure→Kernel→Parameters** and selecting category 3: "Files, Inodes and Filesystems" and changing the value of ETRUNC to 1. The kernel must then be relinked and booted for the new behavior to take effect. Use the **sysadmsh System→Configure→Kernel→Rebuild** selection to relink the kernel.

Verifying System Integrity

The cost of fixing a trusted system that has become untrusted is much greater than the cost of maintaining a trusted system. Once trusted, you can use a few procedures to monitor the integrity of the security perimeter. The programs control the integrity of the Authentication database, the integrity of the system area of the filesystem, and the integrity of the filesystem as a whole.

/etc/fsck

Filesystems containing sensitive files must be considered sensitive entities themselves. Thus, the integrity of the filesystem afforded by the **fsck** program enhances the overall security of the system.

The **fsck** program must be run after any system crash or abnormal termination. As always, make sure the system is in single user mode when running **fsck**. There may be user, system, or audit files in the process of being built when the system crashed. Although that data may be lost, **fsck** can recover some of those files in the *lost+found* directory of the filesystem, and at least fix basic filesystem problems.

To run **fsck**, make the following **sysadmsh** selection:

Filesystems→Check

and specify the filesystem to be checked.

The Audit Trail

Do not overlook the audit trail. It is a valuable resource in tracking down system problems. Not only are significant Protected Subsystem, System Administrator, and Authentication database events recorded there, but the trail also contains those basic system events that can be traced to subsequent system-wide problems.

Checking the System After a Crash

The basic rule is to work from the most basic components of the filesystem outward. Otherwise, corrections made at the higher levels may be undone by programs fixing the lower levels. Given this, use the programs in this section after a system crash or peculiar abnormality in this order:

1. Run a filesystem check.
2. Generate an audit report.
3. Check the consistency of the Authentication database.
4. Check system file permissions.

These programs should be run while the system is in single user (system maintenance) mode.

The Protected Databases

Several databases store the characteristics of the system itself, its users, its administrators, and its subsystems so that a site can control its own security parameters. These databases reside on the system and are maintained by an administrator. The format of these files is discussed in **authcap**(F).

NOTE: The protected databases should never be edited by hand. The trusted system utilities and **sysadmsh**(ADM) selections maintain and display the information contained in the databases. No attempt should be made to modify them through any other means.

The Audit and Device Assignment databases are independent databases. The other databases described below (the Protected Password database, the Terminal Control database, the Subsystem database, and the File Control database) are referred to collectively as the *Authentication database*. The Authentication database is the responsibility of the Authentication Administrator, who has the **auth** authorization. Here are brief descriptions for each of the databases:

Audit — The *Audit* database controls the behavior of the audit system. This includes the types of activity, the system records on the audit trail, the performance/reliability attributes of the audit subsystem, and the file system devices on which audit information is collected. By changing parameters stored in the Audit database, the Audit Administrator can adjust the audit subsystem to suit the performance and security requirements of the site.

Device Assignment — The *Device Assignment* database stores device pathnames which relate to the same physical device. For example, */dev/ttya* and */dev/ttyA* may refer to the same serial port with modem control disabled and enabled respectively. This database is used by **init**(M) and **getty**(M) to stop one form of login spoofing, as described later.

Protected Password — The *Protected Password* database stores security information about each user. The user entry includes the encrypted password (which no longer appears in the regular password database */etc/passwd*) and password change, user authorization, and user audit parameters. By properly setting up this database, the Authentication Administrator controls how users identify and authenticate themselves to the system, the types of privilege users are allowed, and the extent to which users' actions are recorded in the audit trail. The *System Defaults* database, containing the system-wide default security parameters, is considered part of the *Protected Password* database.

Terminal

Access to the system through terminals is controlled by the *Terminal Control* database. This database records login activity through each attached terminal (last login and logout user, time stamps, and so forth). The Terminal Control database lets the Authentication Administrator set different policies on different terminals depending upon the site's physical and administrative needs.

Subsystem

The *Subsystem* database stores the list of users that are given special privilege either to be a subsystem administrator or to perform special functions within a protected subsystem. This database is another element of the Authentication database. It enhances accountability of administrative actions by allowing only specified users to run programs that maintain the internal subsystems. Security is enhanced by controlling who has permission to execute programs that maintain subsystems and by accounting for the real users that assume administrative roles.

File Control

The *File Control* database helps maintain the *integrity* of the Trusted Computing Base. It does this by maintaining a record of the contents and protection attributes of files important to the TCB's operation. This database provides an effective tool for detecting modifications to the active copy of the TCB. The system administrator program **integrity**(ADM) checks the TCB file permissions against this database.

Authentication Database Checking

The **authck**(ADM) program is used to check the consistency of the Authentication Database. There are several options that restrict the scope of the checking. For complete checking, the **-a** flag may be used — perhaps with the **authck** program run from **crontab** or **at**. See **authck**(ADM) for more information.

System Integrity Checking

The **integrity**(ADM) program compares the entries of the File Control database against the actual file permissions on the system. (Access to this program is controlled by the **sysadmin** subsystem authorization, but it is most easily run as the super user.) Files are reported that have more permission levels than are described in the File Control database. When errors are found, you should examine the file to determine:

- What are the owner/group/modes of the file?
- What are the designated permissions of the file? (Use the **-e** option to **integrity**).
- What was the last modify and access time of the file?
- Who was on the system at those times?
- What was the audit trail at those times?

Once the explanation for the discrepancy is found, part of the cleanup must be the resetting of the file to the correct permissions. After that is done for all the files in the report, run the **integrity** program again to validate the permissions.

The **-e** option to **integrity** gives a concise explanation of the cause(s) for the error. The **-m** option reports on those files in the database but not in the system. Sometimes during a system crash or a penetration, significant system files are lost. The *integrity* program is a means to determine this. Note that, for some distributions, some files are not normally present. To know what your system is like as shipped, use the following command:

/tcb/bin/integrity -m

soon after your system is installed and operational. Then, only the additional files reported during normal operation is cause for concern.

The **-v** option of **integrity** causes some extra information to be printed, including reports of files that pass the integrity check. This option produces a lot of output.

All errors found during the integrity check are packaged as audit records that show the audit event as a *Database Event* in the audit trail.

Security-Related Error Messages

This section lists problems and error messages that you may encounter. Each problem is discussed in context, including the reason for the situation, the solution to the problem, and ways to prevent the situation from recurring. The problems described here require logging in as root to fix them. Because the system can potentially lock out all users, (including root) you should create an "override" login for root as described in "Creating an Override Login" in this chapter. If you are locked out and have not established an override tty, you must reset the system or power cycle it, and come up in single-user (maintenance) mode to enter the system. Because shutting off the system can lead to filesystem damage, it is critical that you create an override login.

Login Error Messages

There are several messages associated with logins. In some cases, there are also additional explanatory messages generated by **sysadmsh**. The **sysadmsh** messages are shown in normal rather than courier font to avoid confusing them with login messages. The login messages and their remedies follow:

`Login incorrect`

The user entered their login name or password incorrectly. If this happens repeatedly, you may need to alter the password to permit them to log in again.

When a user (including root) cannot log in and the cause is not related to a locked terminal or account, it is possible that trusted database files are missing or corrupted. If the user sees the "Login incorrect" message and they are certain that their password is correct, it is possible that the */etc/passwd* and/or Protected Password Database has been corrupted. You must use the **Accounts→Examine** selection to determine the real problem. When you enter the username, the account information is checked and more descriptive error messages are generated by **sysadmsh**. These **sysadmsh** messages and the necessary steps remedies are as follows:

There is no entry in Protected Password Database

The database entry located in */tcb/files/auth/[a-z]/username* has been deleted or corrupted. If you find that this is true for a number of users, you should restore the files from backups. For an individual user, the easiest way to repair this is to select the **Identity** information and replace any empty fields and assign a new password.

User exists in Protected Password Database but not in /etc/passwd

Somehow */etc/passwd* was corrupted or someone with **root** powers has made faulty edits to the file. You must log in as **root** and restore your */etc/passwd* file from backups. After doing so, you must also remove the following files to ensure that the system will accept the replaced */etc/passwd* file:

/etc/auth/system/pw_id_map
/etc/auth/system/gr_id_map

If you do not have a backup of your */etc/passwd* file, follow these steps:

1. Move each user's home directory (if it exists) to a temporary name. (For example, move */usr/mattb* to */usr/mattbx*.)

2. Use the **Accounts→User→Examine:Identity** selection for each user and fill in the blanks for group, shell, and the real name of their home directory.

3. Exit **sysadmsh** and move each of the renamed home directories back to their proper names.

```
Account is disabled -- see Authentication Administrator
```

The account is locked for one of three reasons:

1. You have locked the account through the **sysadmsh** selection **Accounts→User→Examine:Logins**. If you want to re-establish the account, use the same selection to unlock the account.

2. The password lifetime for the account has elapsed. The password for the account did not change before the password lifetime was over. To re-enable the account, assign a new password to reset the lifetime. Advise the user that they should change their password before the lifetime expires.

3. A number of unsuccessful tries have been made on the account, surpassing the threshold number you set for locking it. These tries may not have all been made on the same terminal. Before re-enabling the account, it is a good idea to determine the cause for the lock-out. It may be that the user is a poor typist, or another person is trying to lock the account and knows

this is a way to do it, or that a real penetration of the account is being made. You may want to adjust the threshold upward or downward, depending on the nature of the system users, the value of the data, and the accessibility of the system to outsiders.

```
Terminal is disabled -- see Authentication Administrator.
```

The terminal is locked to all users. Similar to account locking, either an Authentication Administrator has locked the account with the **sysadmsh** or a number of incorrect login tries (to one or more accounts) has passed the threshold for that terminal. In both cases, determine what has happened and then use the **sysadmsh** to reset the lock.


```
Account is disabled but console login is allowed.
```

or:

```
Terminal is disabled but root login is allowed.
```

These messages are associated with the super user logging onto the console. Under the assumption that the console device (including a serial console) is a special device and considered a physical resource worth protecting, a lock on the super user account does not prevent the super user from logging into the console. This presents a means for entering the system even when all other accounts or terminals are locked. Before continuing, use the audit trail to investigate the reasons for the lock. A lock-out caused by unsuccessful login attempts on the system console is cleared automatically, but lock-outs due to other reasons remain in effect. The console, in effect, is never locked out for the super user.

```
Cannot obtain database information on this terminal
```

This message can be generated sporadically on a system with a large number of users logging on and off (as on a BBS), but if the message is displayed consistently on all terminals, it is necessary to check the database files.

When database files such as */etc/auth/system/ttys* are updated, a renaming procedure is used to ensure that multiple accesses to the file are managed properly. The contents of the old file (*ttys*) is copied/updated to create the new -t file (*ttys-t*). After that is done the old file (*ttys*) is moved to a -o file (*ttys-o*), and the new file (*ttys-t*) is moved to the original name (ttys). When the above message is displayed consistently, this process has failed and must be corrected. Log in on the override tty and check the */etc/auth/system* directory for *ttys* files. If there are multiple files, the extra files must be removed. However, you must ensure that *ttys* is not empty (for example, if the real *ttys* file is empty and you remove the *ttys-t* file, you are left with an empty *ttys* file). When you have checked the files, do one of the following:

- If *ttys*, *ttys-o*, and *ttys-t* exist then remove *ttys-t* and *ttys-o*.
- If *ttys* and *ttys-t* exists then remove *ttys-t*.
- If only *ttys-t* exists then move *ttys-t* to *ttys*.

```
login: resource Authorization name file could not be allocated due
to: cannot open
Bad login user id
```

The */etc/auth/system/authorize* file has been corrupted or removed. Log in on the override tty and restore the file from backups.

```
Can't rewrite terminal control entry for tty
Authentication error; see Account Administrator
```

It is possible that the */etc/group* file has been corrupted or removed. Log in on the override tty and restore the file from backups. If you did not establish an override tty, reset the system or power cycle it, come up in single-user (maintenance) mode and restore the file. Be sure and set an override tty as instructed in "Creating an Override Login" in this chapter.

Audit Error Conditions

Problem: *The system is currently auditing and a console message indicates that audit is terminating due to an irrecoverable I/O error.*

This typically indicates that a filesystem has been corrupted or a write was attempted on an audit collection that failed due to either lack of space or an I/O error. This situation is irrecoverable and results in the immediate termination of audit.

```
Audit: file system is getting full
```

The audit subsystem may occasionally display this warning when the audit filesystem reaches a certain threshold. This warning message indicates that space is low on a particular device. If additional directories were specified to the subsystem, it automatically switches when the filesystem has reached the threshold value for remaining free space. Otherwise, you (the administrator) must intervene to make more space available. If not, auditing is terminated when the threshold value is reached. The audit daemon program **auditd** may terminate for the same reason. If unable to write a compaction file because of insufficient space or an I/O error, **auditd** terminates. Use the **sysadmsh System→Audit→Disable** selection to terminate auditing if it has not already been done. Analyze the source of the problem and solve it before re-enabling auditing.

```
Authentication database contains an inconsistency
```

This message is displayed while running one of the programs associated with the TCB or a protected subsystem. The Authentication database integrity is in question. The Authentication database is a composite of the Protected Password database, the Terminal Control database, the File control database, the Command Control database, the Protected Subsystem database, and the System Defaults file, and the message applies to all of these. Either a data entry is not present when expected, or the items within an entry are not correct. This message is inherently vague and is meant to be. The invoking user is alerted to the problem but not given enough information about the cause to allow them to exploit an integrity problem within the security perimeter. The real reason for the problem may be found in the audit trail, if *Database Events* were enabled for the user that generated the message.

Authorization Problems

```
You do not have authorization to run ....
```

The command is part of a protected subsystem. For that subsystem, the Authentication Administrator has not provided you with the kernel authorization needed to run this command and/or related commands. The Authentication Administrator uses the **Accounts→User→Examine→Authorizations** selection to grant or deny such authorizations.

Problem: *A command you are running is not providing all the information you seek or you cannot perform some actions. You know there is more data to receive or request.*

The command may be part of a protected subsystem. Although you are not totally shut out from the command, you cannot use all the options or see all the data. As above, the Authentication Administrator needs to grant additional authorizations.

Adding Dial-in Password Protection

If desired, you can define special dial-in passwords on selected tty lines, requiring selected classes of users to input dial-in passwords. Logging information, including the last time of connection, can be stored for later use.

Specific dial-in lines that require passwords are defined in the file */etc/dialups*. The format is one tty device name per line, for example:

```
/dev/tty1A
/dev/tty5C
```

The actual dialup passwords are kept in the file */etc/d_passwd*. The password format is the same one used in */etc/passwd*, The first field ("user name") in */etc/d_passwd* is not a user name, but the name of a shell program (for example, */bin/sh*) used in */etc/passwd*. If the login shell of the user attempting to log in (on a tty line listed in */etc/dialups*) is listed in */etc/d_passwd*, then the user is prompted for the dial-in password stored in */etc/d_passwd*.

Here is the syntax for creating a dial-in password:

```
passwd -d dialname
```

Change the password for dialup shell *dialname* (listed in */etc/d_passwd*). If *dialname* begins with a slash (" / ") the entire shell name must match. Otherwise the password for every shell whose basename is *dialname* is changed. Only the super-user may change a dialup shell password.

Chapter 6
Backing Up Filesystems

The main task of a system administrator is to ensure the continued integrity of information stored on the system. Files and filesystems can be damaged and data lost in the following ways:

- Power interruptions (make certain you have a surge protector)
- Hardware failures (particularly the hard disk)
- User errors (accidental removal of important files).

The importance of having up-to-date backups cannot be overstated. If your system has a number of active accounts, backups require daily attention. It is difficult to estimate the magnitude of a simple loss of data until an accident occurs and several weeks or months of work is gone in an instant.

A filesystem backup is a copy, on storage media (floppy disks or tape) of the files in the root filesystem and other regularly mounted filesystems (for example, the */u* filesystem). (See the "Using Filesystems" chapter in this Guide for a discussion of filesystems.) A backup allows the system administrator (or user with the **backup** authorization) to save a copy of a filesystem as it was at a specific time.

Strategies for Backups Using sysadmsh

As system administrator, you should familiarize yourself with this chapter and create a schedule as instructed. When this schedule is complete, you have only to insert a media volume and respond to a series of prompts to perform your daily backups.

The primary purpose of the **sysadmsh** filesystem backup selection is to provide a dependable schedule of filesystem backups for systems with many users and large filesystems. The program automatically locates modified files and copies them to backup media. If your system has many users and a large number of files that are modified daily, the "scheduled" backup option uses a predefined schedule to make regular backups. When the Backups selection is invoked, the program presents each task as a menu option. To perform a task, simply choose the appropriate option from the menu and supply any required information.

For backups of a more informal nature, **sysadmsh** includes an option for "unscheduled" backups. This allows the system administrator to perform a single, complete backup of a filesystem. (Note this type of backup covers the entire filesystem, not just modified files, and may require a number of storage media volumes.) If you intend to rely on unscheduled backups, be sure and perform one at least once a month.

Floppy Drive Backups and Large Systems

If your system has only a floppy drive, backups for large systems with several users can be time-consuming and use a great deal of media. A complete backup of a 20 megabyte filesystem requires *fifteen* 1.2 MB 96tpi diskettes, while a single 450 foot cartridge tape can store more than twice that amount. More importantly, diskettes require the presence of the operator to keep feeding floppies, whereas a single cartridge tape can be inserted and the operator need not remain by the system. If your system has a large number of users and just a floppy drive, you are advised to install a cartridge tape drive, or make complete system backups once per week and warn your users to make individual backups of their own files on a regular basis.

Preparations for Scheduled Backups

The only mandatory requirement for scheduled backups is the creation of a backup schedule. In addition, it is recommended that the system administrator follow the optional procedures for labeling, storing and logging backups. A detailed explanation of backup levels is included at the end of this chapter in case it is necessary to design a more specialized schedule.

Creating a Backup Schedule

The first step is to create a timetable for backups using the **schedule** file. The file is located in */usr/lib/sysadmin*. It contains all the data needed for the system to perform a system backup, including:

- The name of your site or machine
- The media type and drive to be used
- A precise schedule of filesystems to be backed up.

The sections that follow explain what changes should be made to the **schedule** file provided with your distribution.

Edit the schedule File

You can return to the desktop and double-click on the Sysadmsh icon to bring up the main **sysadmsh**(ADM) menu. Then make the following selection:

Backups→Schedule

(**sysadmsh** uses the **vi**(C) editor by default, but you can set the SA_EDITOR environment variable to the editor you prefer. See **environ**(M) or **sh**(C) for an explanation of how to set environment variables.) The subsections that follow explain the exact changes you need to make to this file.

```
# SYSTEM BACKUP SCHEDULE
site mymachine
# Media Entries
#
# 48 tpi 360K floppy 0
# media /dev/rfd048ds9 k 360 format /dev/rfd048ds9
# 48 tpi 360K floppy 1
# media /dev/rfd148ds9 k 360 format /dev/rfd148ds9
# 96 tpi 720K floppy 0
# media /dev/rfd096ds9 k 720 format /dev/rfd096ds9
# 96 tpi 720K floppy 1
# media /dev/rfd196ds9 k 720 format /dev/rfd196ds9
# 96 tpi 1.2 MB floppy 0
media /dev/rfd096ds15 k 1200 format /dev/rfd096ds15
# 96 tpi 1.2 MB floppy 1
# media /dev/rfd196ds15 k 1200 format /dev/rfd196ds15
# 135 tpi 720K floppy 0
# media /dev/rfd0135ds9 k 720 format /dev/rfd0135ds9
# 135 tpi 720K floppy 1
# media /dev/rfd1135ds9 k 720 format /dev/rfd135ds9
# 135 tpi 1.44 MB floppy 0
# media /dev/rfd0135ds18 k 1440 format /dev/rfd0135ds18
# 135 tpi 1.44 MB floppy 1
# media /dev/rfd1135ds18 k 1440 format /dev/rfd135ds18
# Cartridge tape 1
# media /dev/rct0 k 60000 125000 150000 tape erase
# Mini cartridge drive (10MB)
# media /dev/rctmini k 8800 format /dev/rctmini
# Mini cartridge drive (20MB)
# media /dev/rctmini k 17200 format /dev/rctmini
# Mini cartridge drive (40MB)
# media /dev/rctmini k 37500 format /dev/rctmini
# 9-track tape drive
# media /dev/rmt0 d 1600 2400 1200 600

# Backup Descriptor Table

#   Backup   Vol.   Save for     Vitality        Label
#   level    size   how long     (importance)    marker
    0        -      "1 year"     critical        "a red sticker"
    1        -      "4 months"   necessary       "a yellow sticker"
    8        -      "3 weeks"    useful          "a blue sticker"
    9        -      "1 week"     precautionary   none

# Schedule Table
#              1 2 3 4 5   6 7 8 9 0   1 2 3 4 5   6 7 8 9 0   Method
# Filesystem   M T W T F   M T W T F   M T W T F   M T W T F
  /dev/rroot   0 x 9 x 9   8 x 9 x 9   1 x 9 x 9   8 x 9 x 9   cpio
```

Figure 6-1. The schedule File

Add the Name of Your Site or Machine

Simply change the *mymachine* entry at the top of the file to the name you wish.

Select the Media Device that Matches Your Configuration

The default **schedule** file appears as in Figure 6-1. The 96tpi 1.2 Megabyte floppy drive 0 is the default drive (reproduced below). The pound signs (#) are comment symbols used to "comment out" text so that it is ignored by the program. Note that the default drive is the only one without a comment symbol. If you plan to use a drive other than the default, put a comment symbol in front of the 96tpi drive and remove the comment symbol from in front of the drive you wish to use. The remaining drives should remain commented out.


```
# 96 tpi 720K floppy 1
# media /dev/rfd196ds9 k 720 format /dev/rfd196ds9
# 96 tpi 1.2 MB floppy 0
media /dev/rfd096ds15 k 1200 format /dev/rfd096ds15
# 96 tpi 1.2 MB floppy 1
# media /dev/rfd196ds15 k 1200 format /dev/rfd196ds15
```

Figure 6-2. Default Media Entry

Edit the Backup Descriptor Table

Directly below the media drive lines is the Backup Descriptor table. This table, reproduced in Figure 6-3, describes each backup level in terms of volume size, how long it is to be stored, how important it is, and how it is marked. The default entries should prove useful, but the volume size entries must be edited according to the type of media you are using.

If you are using floppy disks, leave the dashes in the "Vol. size" column as they are. This causes the backup program to take the volume size from the media entry for that device.

If you are using tapes or tape cartridges, replace each dash in the "Vol. size" column with the size (in kbytes) of the tape volume. If you are using tapes that are all the same size for each backup level, replace each with the size of the tape you're using.

The last column contains label entries that are discussed in "Labeling Your Backups" later in this section.

```
#   Backup   Vol.   Save for      Vitality          Label
#   level    size   how long     (importance)      marker
    0        -      "1 year"     critical          "a red sticker"
    1        -      "4 months"   necessary         "a yellow sticker"
    8        -      "3 weeks"    useful            "a blue sticker"
    9        -      "1 week"     precautionary     none
```

Figure 6-3. Backup Descriptor Table

Edit the Backup Schedule Table

The default schedule assumes that backups will be done every other day. A precise understanding of backup levels is not critical to using the schedule. Level 0 is the lowest level backup. It backs up everything on the filesystem, while 1, 8, and 9 each back up only the files that have changed relative to the last lower-level backup. (For a complete discussion, see "An Explanation of Backup Levels" at the end of this chapter.) The example **schedule** files in this chapter includes an entry for a */u* filesystem that is not present in the default file. Note that there is a backup done every other day for the root filesystem and once a day for the */u* filesystem. This is because the */u* filesystem (user accounts) changes much more frequently than the root filesystem, which contains the system files.

If you do not have a */u* filesystem, then your user accounts are located in the root filesystem (in the directory */usr*). If this is so, the schedule table is pre-configured to back-up the root filesystem. However, if you have added a */u* filesystem, edit the schedule table and add an entry for */dev/ru*, as shown in Figure 6-4. This ensures that backups will be made of the additional filesystem. If you do not have a */u* filesystem, but you do want daily backups, this entry can also be modified and used for the root filesystem.

```
#              1 2 3 4 5   6 7 8 9 0   1 2 3 4 5   6 7 8 9 0   Method
# Filesystem   M T W T F   M T W T F   M T W T F   M T W T F
  /dev/rroot   0 x 9 x 9   8 x 9 x 9   1 x 9 x 9   8 x 9 x 9   cpio
  /dev/ru      9 0 9 9 9   9 8 9 9 9   9 1 9 9 9   9 8 9 9 9   cpio
```

Figure 6-4. Backup Schedule Table

Note that the Monday-Friday notation can be misleading; if a backup is postponed or unsuccessful (because of bad media, for example) then that same level backup is attempted again at the next scheduled backup. This offsets the schedule, but does not alter the established sequence of backups. The numbered scale of 1-0 above M-F is more accurate, but less useful to people, who work in day and week units.

In addition, if you add lines for other filesystems, you should take care not to schedule two level 0 backups of large filesystems on the same day; the process will be lengthy and may slow your machine significantly.

Labeling Your Backups

It is important to label your backup tapes with meaningful and accurate information. If your backups consist of a pile of haphazardly labeled tapes, it will be difficult to locate data at a later date.

Figure 6-5 is a suggested format for media labels.

Name of computer	Backup level Filesystem Name save until date # of blocks on volume	Date made
Name of backup person		volume # of #

Figure 6-5. Sample Media Label

The date on the label, and the date from which you calculate the "save until" date, should be the date of the business day covered by the backup. This is to avoid confusion if it becomes necessary to restore information from this tape.

You may have noticed that the **schedule** file has a proposed color-coding scheme for easy reference, as emphasized in Figure 6-6.

```
#   Backup   Vol.   Save for     Vitality        Label
#   level    size   how long    (importance)    marker
    0        -      "1 year"    critical        "a red sticker"
    1        -      "4 months"  necessary       "a yellow sticker"
    8        -      "3 weeks"   useful          "a blue sticker"
    9        -      "1 week"    precautionary   none
```

Figure 6-6. Backup Labeling Scheme

If there is more than one tape for a single backup, mark the date label on each volume to indicate the volume number and number of volumes, such as "1 of 2" and "2 of 2" for a two volume backup. Finally, place a label on the side of the box or enclosure marked with the name of the computer, the filesystem, and the backup level completed.

Keeping a Log Book

It is recommended that a written log book be maintained for each computer. In addition to maintenance information (such as when breakdowns occur and what was done about it), you should record the following information:

Date	Just as with the tape label, this date should be the last day covered by the backup.
Filesystem	The name of the device backed-up on the current tape.
Backup level	The backup level of the current tape.
#Vols	Number of tape volumes.
#Blocks	Number of tape blocks. This information is output when the backup is completed. (Pay attention to this figure; if the number of blocks for a level 9 backup consistently exceeds four digits for a particular filesystem, then you should probably increase the frequency of backups for that filesystem to lessen the burden.)
Start/finish time	(Optional.) The time from the start of a backup of a filesystem until the last error check is completed. The times are displayed after the backup is finished. The finish time will often be inaccurate, since you may be out of the room when the backup finishes, and the machine sits idle before you return.

If there are problems with the backup, record these in the log book as well, including any error messages that come to the screen.

Performing a Scheduled Backup

This section describes how to perform a backup using a defined schedule. Do not attempt this until you have edited (or at least examined) the schedule file to make certain that it suits your needs.

The system administrator should schedule backups at times when few (if any) users are on the system. This ensures that the most recent version of each file is copied correctly.

A regular schedule of backups requires a good supply of media and adequate storage for them. Level 0 backups should be saved at least a year, longer if they are important. Lesser backups should be saved at least two weeks. Media volumes should be properly labeled with the date of the backup and the names of the files and directories contained in the backup. After a backup has expired, the media may be used to create new backups.

Using Formatted Media

If you use media that requires formatting, such as floppy disks or mini tape cartridges, you may wish to format several volumes before you begin. The exact number of volumes depends on the number and size of files to be backed up. For details on how to format your media, see the "Using Floppy Disks and Tape Drives" chapter in this guide. You also have the option to do formatting from the **sysadmsh** program. (Note that rctmini tape cartridges take a very long time to format.)

Starting the Backup

To run your scheduled backup, follow these steps:

1. First, return to the desktop and double-click on the Sysadmsh icon to bring up the main **sysadmsh**(ADM) menu. Then make the following selection:

   Backups→Create→Scheduled

2. A menu is displayed that looks like the following:

```
Level 0 backup of filesystem /dev/rroot, 22 Sep 1989
        tape size:      1200 Kb
        tape drive:     /dev/rfd096ds15
This tape will be saved for 1 year, and is critical.

M)ounted volume, P)ostpone, C)heck or F)ormat volumes,
                                        R) Retension or H)elp:
```

   The media type displayed is the one entered in the **schedule** file. Load a volume, tape or disk, into the selected drive. Enter "m" to tell the program the volume is mounted, and press **Return**.

3. the system displays the current date and the date of the last backup:

```
Level 0 backup of filesystem: /dev/rroot
Backing up all files
Generating list of pathnames for backing up ...
```

   This process will take a few minutes.

4. The system then begins to copy files to the drive. If a volume runs out of space, the program displays the following messages:

```
Reached end of medium on output
Insert volume 2 and press <RETURN> to continue or 'q' to exit.
```

   NOTE: If you are using 5.25-inch floppies for your backups, make certain you close the floppy door before pressing **Return.**

   Remove the present volume, insert a new volume, then press **Return.** The program continues to copy files to the new volume. Repeat this step until the program displays the message:

```
n blocks
Check critical volumes for format errors
```

5. If an error occurs, the backup is declared unsuccessful and is retried from the beginning. Your media could be bad, so replace it if errors persist. The menu appears as follows:

```
M)ounted which volume, E)rror on previous volume, D)one,
S)kip checks, or H)elp:
```

After the backup has been successfully performed, instructions are given on how to label the volumes. If you are checking the format, make certain you insert the first volume as instructed, or the backup will abort. If you don't want to check the volumes, select "Skip". Make certain that you write-protect your volumes.

Performing an Unscheduled Backup

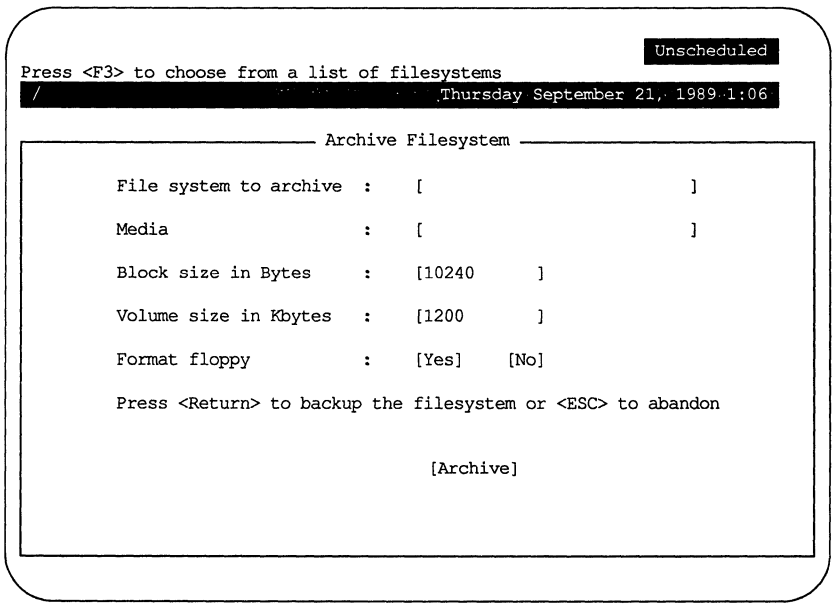
You can create backups on tape or disk. If you use media that requires formatting, such as floppy disks, you may wish to format several volumes before you begin. The exact number of volumes depends on the number and size of files to be backed up. For details on how to format media, see the "Using Floppy Disks and Tape Drives" chapter in this Guide. You also have the option to do formatting from within **sysadmsh**.

To create an unscheduled backup, follow these steps:

1. First, return to the desktop and double-click on the Sysadmsh icon to bring up the main **sysadmsh**(ADM) menu. Then make the following selection:

   Backups→Create→Unscheduled

2. The following form is displayed:



3. Select the filesystem to backup by entering the name or pressing **F3** to get a point-and-pick list. The menu lists all filesystems found in the file */etc/default/filesys*. (See **filesys**(F).) Use the arrow keys to select the filesystem you wish to back up and press **Return**.

4. Next, select the media device to be used by entering the name or pressing **F3** to get a list. The block size is selected automatically.

   NOTE: Take care when selecting the number of the media device. For example, make certain that you don't select "Floppy Drive 1" (the secondary floppy drive) when you want "Floppy Drive 0" (the primary floppy drive). If you make this error, the backup is aborted and you must start over.

5. You can format as many volumes as you wish by inserting them into the drive and selecting "Yes" on the *Format floppy*. As discussed earlier, mini-cartridge tapes can also be formatted.

6. Load a volume, tape or disk, into the selected drive, and press **Return**. The system displays the current date and the date of the last backup. The system then begins to copy files to the drive. If a volume runs out of space, the following is displayed

```
Reached end of medium on output
Insert volume 2 and press <RETURN> to continue or 'q' to exit.
```

7. Remove the first volume, insert a new volume, then press **Return**. The program continues to copy files to the new volume. Repeat this step until the program displays the message:

   DONE

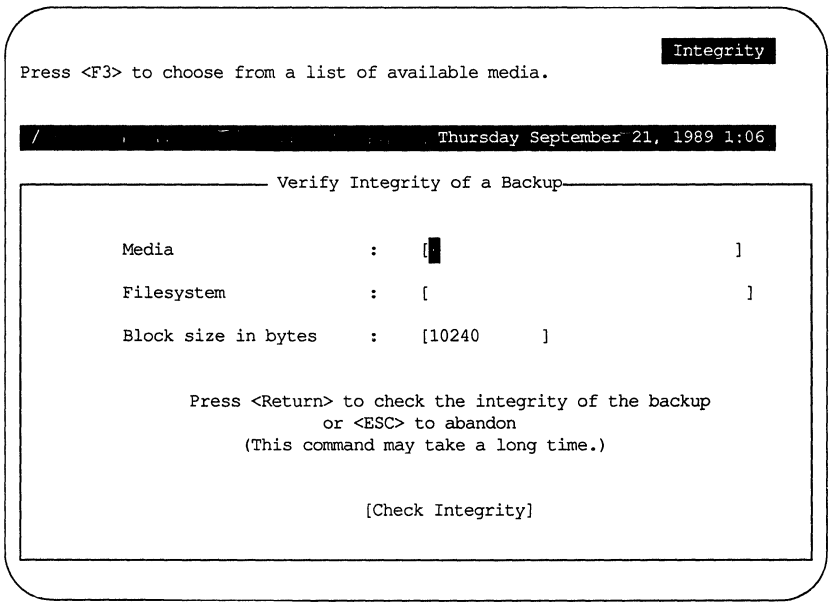
If you are using floppies, you may need to repeat the last step several times before the backup is complete. You should label each volume as you remove it from the drive. For example, label the first volume "Volume 1,', the second "Volume 2," and so on.

Verifying a Backup

To ensure that your backup volumes are accurate and error-free, the **sysadmsh** backup menu includes an Integrity option. The volumes are checked to see if they are readable and the contents are listed.

First, return to the desktop and double-click on the Sysadmsh icon to bring up the main **sysadmsh**(ADM) menu. Then make the following selection:

Backups→Integrity



Enter or select the media type and insert each volume of the backup in turn. This is a lengthy process; a large backup will take some time.

Getting a Backup Listing

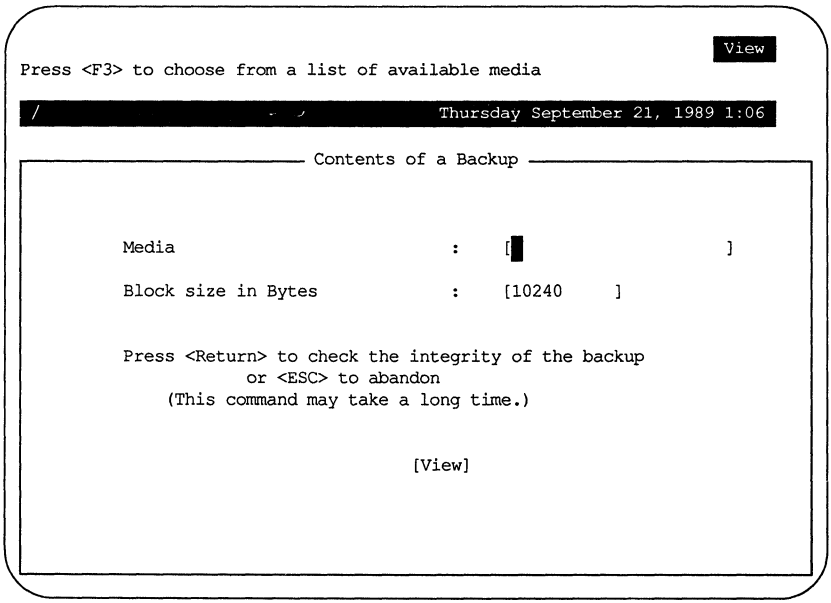
You can examine a list of the files you have backed by generating a listing from the **sysadmsh** Backups Menu.

To get the listing, follow these steps:

1. First, return to the desktop and double-click on the Sysadmsh icon to bring up the main **sysadmsh**(ADM) menu. Then make the following selection:

   Backups→View

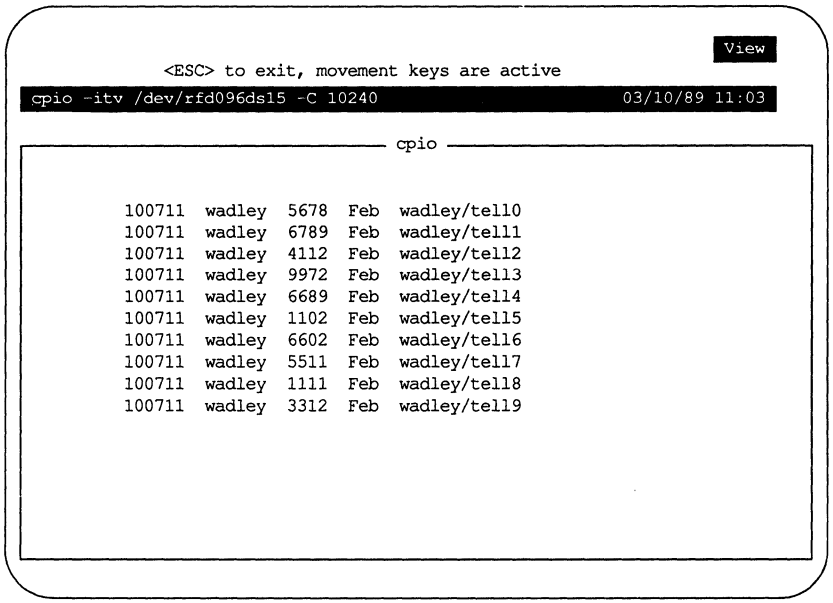
2. The following forms is displayed:



3. Press **F3** at the first field to get a listing of media devices. The block size is selected automatically.

4. The program prompts you to insert the first backup volume. Load the first volume, then press **Return**.

5. When all volumes of the backup are read, a screen similar to the following is displayed:



Restoring Individual Files or Directories from Backups

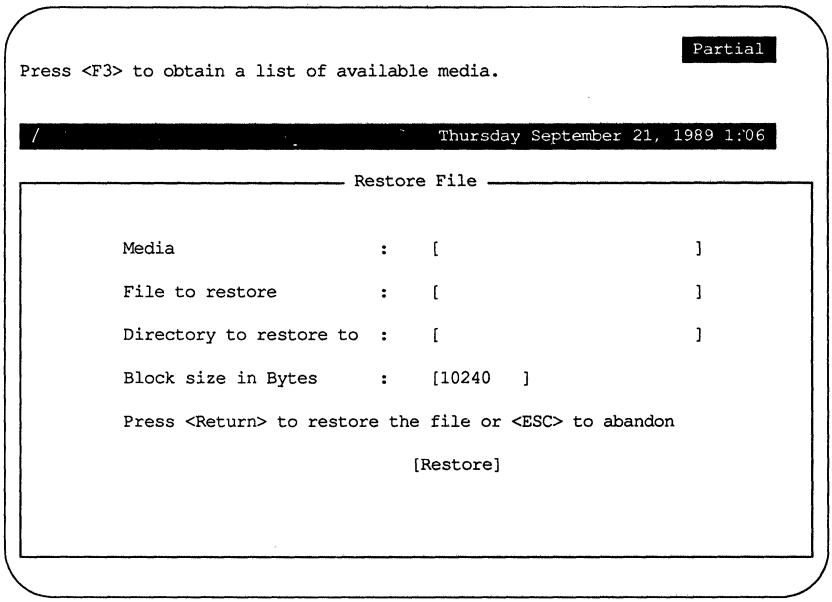
You can restore individual files or subdirectories from your filesystem backup volumes by invoking **sysadmsh**. You will need the complete set of backup volumes containing the latest version of the file or files you wish to restore. If you are restoring a file that has not been changed recently, use the last level 0 backup.

To restore a file, follow these steps:

1. First, return to the desktop and double-click on the Sysadmsh icon to bring up the main **sysadmsh**(ADM) menu. Then make the following selection:

   Backups→Restore→Partial

2. You see the following:



3. Press **F3** first to select the Media type from a point-and-pick list. When selected, a window pops up to confirm the drive is ready:

```
Please make sure the media is in the drive and
the drive is on line.

Press <Return> to format the disc or <ESC> to abandon
```

4. Load volume 1 of the backup set into the drive, then press **Return**. When this request is satisfied, you are returned to the "Restore File" menu. Enter filename next, then press **Return** to move to the *Directory* field, entering the directory you wish to restore the file(s) to.

   NOTE: Two important points:

   - When specifying the pathname, the leading slash (/) must be removed. For example, if you are restoring the file */bin/foo*, you must specify it like this:

     bin/foo

   - If you respond with the pathname of the original location, the restored files will overwrite any files by the same names in that location. It is important to be sure that the files on the backup are the desired versions of these files. If you are not absolutely sure that your backup contains the preferred version of the files, you should restore them to a temporary location, such as */tmp*, and compare them with your current files on disk using **diff**(C) or **cmp**(C).

5. Now the actual command line used is displayed, as in the following example using **cpio**:

   ```
   cd /tmp; cpio -iudv -I/dev/rfd096ds15 -C 10240
   ```

6. The archive is searched for the files specified and the filename is displayed after it is restored to the specified locations on your hard disk. You are also prompted to switch volumes if necessary. If you know all the files you want have been restored, you can delete out of the restore using the **Del** key. (The program continues to search to the end of the backup.)

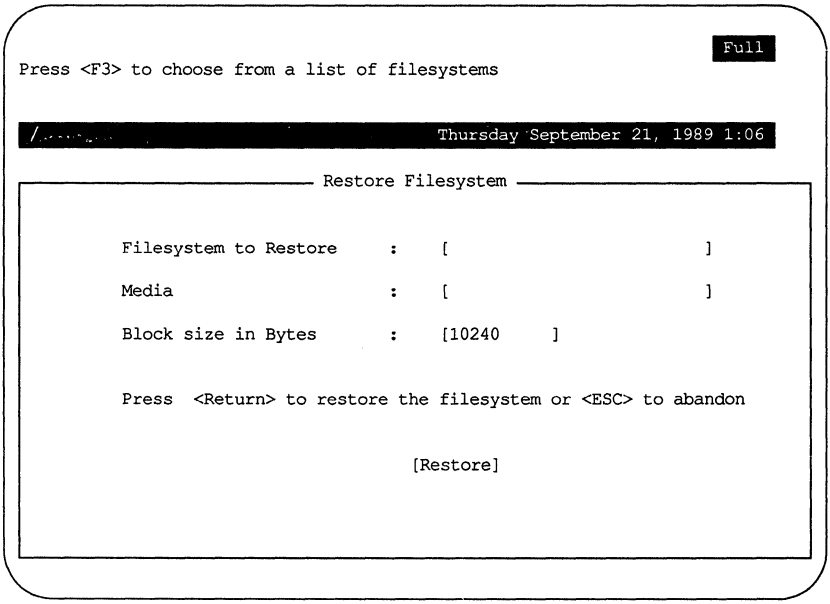
Restoring an Entire Filesystem

Follow these steps to restore your filesystem backup:

1. Insert the first volume, return to the desktop and double-click on the Sysadmsh icon to bring up the main **sysadmsh**(ADM) menu. Then make the following selection:

   Backups→Restore→Full

   The following form is displayed:



2. Enter the name of the filesystem, or press **F3** for a point-and-pick list. Do the same for the media device. You are asked to confirm this is what you wish to do.

3. Now the actual command line used is displayed, as in the following example of restoring a */u* filesystem using **cpio**:

```
cd /u; cpio -iudv -I/dev/rfd096ds15 -C 10240
```

4. As each file is restored, the name is printed on the screen. If your backup has multiple volumes, you are prompted to insert each in turn:

```
Reached end of medium on input
Change to part n and press <RETURN> key. [q]
```

When the restoration process is complete, the number of blocks restored is displayed.

An Explanation of Backup Levels

The most straightforward and dependable way to ensure the safety of data is to back up everything on a filesystem at one time. However, filesystems can be large (as much as 400 MB or more), and may take hours to backup. The concept of backup levels (or incremental backups) addresses this problem. The general idea of an incremental backup is to back up only those files that have changed since a previous backup. This can significantly reduce the size and duration of the backup. Consider the following scheme:

Monthly	complete backup
Weekly	everything newer than last week
Daily	everything newer than yesterday

This means that at the end of every month, the entire filesystem is backed-up. Each week, the files that have changed since last week are backed-up, and each day, any files that have changed since yesterday. If at some point a filesystem is damaged, you would simply restore the last full (monthly) backup, the last weekly backup, and any daily backups that happened just prior to the accident. Thus it is always possible to reconstruct a filesystem from a series of backups.

While this is a simple method to understand, the implementation using incremental backup levels is not.

Principles of Incremental Backup Levels

To make the business of backing up files more efficient, the backup facility uses a progressive series of levels, each of which is based on the last occurrence of a lower level backup. Up to ten different levels of backups are supported, giving the system administrator tremendous flexibility in organizing backups.

Level	Files Saved
0	all files on the filesystem
1	files changed since last level 0 backup
2	files changed since last level 1 backup
3	files changed since last level 2 backup
-	- -
9	files changed since last level 8 backup

The full ten levels would be used to accommodate computers with massive filesystems; average systems will use only a few levels. The levels serve to subdivide a backup into manageable units. It is important to realize that each backup level creates backups based on the previous (next lowest) level backup. This means that the order of the backups is not significant, but the level number is.

For example, let's assume that the following backups were done for a week:

Day	Level	Files Backed Up
Mon	0	All files on filesystem
Tue	5	All files changed since Monday
Wed	2	All files changed since Monday
Thu	7	All files changed since Tuesday
Fri	5	All files changed since Wednesday

This example is illogical, but serves to demonstrate how the levels work. Remember that each of the backups saves the files changed since the next lower level backup, and that level 0 is the lowest. Therefore, the level 5 on Friday backs up all files changed since the next lowest number, level 2, on Wednesday. The level 5 on Tuesday saves only those files that have changed since the day before, since the only previous lower level backup is a 0. If all the backup levels except Monday were level 5, each would still backup all files that changed since the level 0 on Monday.

How the Default Schedule Works

The default **schedule** file provided with your distribution uses only four levels, and is optimized for use on systems under moderate use (8-10 users with total disk storage of 200-400 MB). This default schedule, with an additional line for */u*, is shown in Figure 6-7.

```
#                1 2 3 4 5   6 7 8 9 10   1 2 3 4 5   6 7 8 9 10
#  Filesystem    M T W T F   M T W T F    M T W T F   M T W T F
   /dev/rroot    0 x 9 x 9   8 x 9 x 9    1 x 9 x 9   8 x 9 x 9
   /dev/ru       9 0 9 9 9   9 8 9 9 9    9 1 9 9 9   9 8 9 9 9
```

Figure 6-7. The Default Schedule

The /u Filesystem

Filesystem */dev/u* is a heavily-used resource. Some level of backup is performed every day. This scheme is designed to minimize resources while maximizing safety; if one or more of the backups for that week is lost or goes bad, there is sufficient redundancy to minimize any loss of data.

According to the default schedule, a full (level 0) backup of */dev/ru* occurs at the beginning of the month. (Because a level 0 is done on the root filesystem on Monday, the level 0 for */u* is done on Tuesday.) Wednesday, a level 9 backup saves just those files on */dev/ru* which have changed since the level 0 backup. By the end of the week far fewer floppies or tapes are used than the number needed for full backups each day. Time is substantially reduced as well. If it is necessary to restore the filesystem to the last recorded state, you would restore the last level 0 backup, followed by each of the most recent lower-level backups that have been done since.

Note that each Tuesday, a lower level backup (0, 1 or 8) occurs that saves everything since the beginning of the month and causes each of the level 9 that follow it to be based on that week. This way the level 9 backups don't become too large and redundant.

The root Filesystem

The root filesystem contains the operating system and other system files. It changes less frequently, so it is not backed up every day. Each Monday, a lower level backup is done, and level 9 backups are done twice per week. Just as with the */u* filesystem, the level 9 backups are restricted to cover only those files that have changed during that week.

How Backups are Used to Restore a Filesystem

Now, let's assume you have a hardware failure that ruins the information on the hard disk. Assume it happens on the last Thursday of the month, just before the backup was to be done that evening. You fix the hardware problem and reinstall your system, but how do you restore your backups? Restore the last occurrence of each backup level, in ascending order:

- level 0 (done on the first Tuesday of the month)
- level 1 (done on the third Tuesday)
- level 8 (done on the fourth Tuesday)
- level 9 (done on Wednesday evening)

You wouldn't need to restore the level 8 that was done on the second Tuesday, because the level 1 that followed it covered the same files. The only information that is missing is what was changed during the day on Thursday, just before the crash. This is the primary reason for backups; recovery should be straightforward and with a minimum of loss.

Chapter 7

Adding Device Drivers with the Link Kit

This chapter explains how to add device drivers to the UNIX kernel, a process known as "linking". To change any component of the UNIX kernel, it is necessary to use the Link Kit to relink the kernel. The Link Kit consists of a set of kernel components in the form of relocatable object modules plus various programs and shell scripts used to link the components together.

The most common use for the Link Kit is to add device drivers to the system. A device driver is the software interface between a peripheral device and the operating system. Each device that can be used with the system must have a device driver. New drivers are generally supplied when adding a peripheral device to the system; they must be configured into the kernel before the device will function.

Device Drivers

A UNIX device driver is a set of routines that communicate with a hardware device, and provides a means by which the operating system can control the device in order to perform Input/Output (I/O) operations.

A device driver is usually supplied as a single software module. Installing this software into the kernel is as important as the actual hardware installation. It must be completed before the device can be used. A driver is usually accompanied by an auxiliary program or shell script that helps to form the links between driver and kernel.

To prepare for installing a new device driver:

- After shutting down the system and switching the power off, install the hardware device on the system according to the manufacturer's instructions.
- Boot the system and enter system maintenance mode. All the operations described as part of the installation process are performed in this mode.

- Make sure the Link Kit is installed. If it is not already installed, install it using the **custom**(ADM) command.
- Change to the directory containing the Link Kit so you can run the configuration tools.

  cd /etc/conf/cf.d

  Most of the following installation procedures are performed in this directory.

Installing Device Drivers

The exact instructions for installing a new device driver are different for each type of device. Read the specific installation instructions that are provided with the device driver software.

After the Link Kit is installed and the instructions read, the next step depends on how much of the work has already been done by the driver's vendors.

Many software vendors provide automatic driver installation utilities compatible with **custom**. insert the vendor's floppy in the floppy drive and enter:

custom

Δ **sysadmsh** users select: System→Software

Select the option to add a supported product, and follow the instructions that appear on the screen. **custom** should run any UNIX System V-compatible, automatic installation software provided with the driver. This installs the device driver software and links a version of the UNIX kernel that contains the new device driver. After **custom** completes, the next step is usually to test the newly created kernel. See the device driver documentation for details.

If the driver is *preconfigured*, follow the instructions in "Installing Preconfigured Drivers". Otherwise, proceed to "Installing Drivers Without Configuration Shell Scripts" to determine the commands necessary to configure the driver.

Installing Preconfigured Drivers

The driver installation floppy may come with a shell script to include the new driver. If such a script is present, run it by entering:

./scriptname

where *scriptname* is the name of the script. Most scripts also create all necessary device special files; if this is the case, shutdown the system and boot the kernel that now includes the new driver. If your script does not create the proper special files in */dev*, you must create them with the **mknod**(C) command. For more information on making device special files, see Step 10 in "Installing Drivers Without Configuration Shell Scripts" or refer to the **mknod**(C) manual page.

Installing Drivers Without Shell Scripts

If no configuration shell script is present, you should follow the steps below (if you have problems, contact the driver vendor for help):

1. Make a backup copy of the kernel with the following command:

   cp /unix /unix.old

2. Get the names of the driver routines from the driver module. The driver module is the *.o* file (usually *Driver.o*) from the installation media. Enter:

   ./routines Driver.o

   This command can take several minutes. Write down the names produced. Most of these names are either configurable driver routines or driver priority levels. Some names may be spurious.

   NOTE: If you see several *.o* files, the installation media contains more than one driver. Each *.o* file is a driver module. The names of the files for each driver probably contain a prefix, which is usually the name of the device. For example, a driver module for a serial I/O device might be named *sioDriver.o*. You must repeat Steps 2-10 of the procedure described here for each driver you want to install.

3. Find the *interrupt priority level.* Driver priority levels have names consisting of the string *spl* followed by a number between 0 and 7. The largest number following the *spl* string is the interrupt priority level. For example, if the name *spl6* is the highest priority level produced, the device's interrupt priority level is 6. Then, cross all *spl* routines off the list.

4. Find the relevant driver routine names. Configurable driver routines all have a common prefix, such as *sio*. Each prefix is followed by one of a small group of suffixes: *open, close, read, write, ioctl, startup, exit, fork, exec, init, halt, poll, strategy, print, _tty,* or *intr*. If any routine name does not fit this pattern, cross it off the list. For example, running **routines** on *sioDriver.o* produces a long list of routines that begin with *sio*, and a single *ttinit* routine. In this case, you would cross out *ttinit* because it doesn't begin with *sio*. The *sio* driver contains a few other routines that would also be crossed out, such as *siopinit* because of the extra "p". *sio* is an extreme case: most drivers do not have spurious routine names scattered throughout the relevant ones.

5. Determine whether the peripheral is a *block* device or a *character* device. If any routine ends with *strategy* or *print*, it is a block device. If any routine ends with *read*, *write*, or *ioctl*, it is a character device. A peripheral can be both a block and a character device. If none of these routines are present, consider the peripheral to be a character device.

6. Create a subdirectory in */etc/conf/pack.d* to hold the driver package you are installing. Use the common prefix of the configurable driver routines (such as *sio*) as the name of the subdirectory.

   mkdir /etc/conf/pack.d/*prefix*

   If you plan to use a device name that is different from the prefix of the routines when you configure the driver (see the **-h** option to the **configure** command described later), use that device name as the name of the subdirectory instead of using the prefix.

   Move the files related to the particular driver to the new subdirectory. These files should include at least a *Driver.o* file. If the driver package also contains *space.c* and *stubs.c* files, move these files as well.

   mv Driver.o space.c stubs.c /etc/conf/pack.d/*prefix*

   NOTE: If the files you extracted from the installation media contained more than one driver (several *.o* files), the names of the files for each driver package are probably prefixed with the device name (such as *sioDriver.o*, *siospace.c*, and *siostubs.c*). When you move the files to

the subdirectory of */etc/conf/pack.d*, remove the prefix from the filenames so that the files are called simply *Driver.o*, *space.c*, and *stubs.c*. For example:

mv sioDriver.o /etc/conf/pack.d/sio/Driver.o
mv siospace.c /etc/conf/pack.d/sio/space.c
mv siostubs.c /etc/conf/pack.d/sio/stubs.c

Remember to create a subdirectory and move the related files for each driver you are installing.

7. Obtain the *major device number* with the following command and write it down for later use:

   ./configure -j NEXTMAJOR

8. Select *interrupt vectors* for the device. If a routine containing the name *intr* exists, refer to the device's hardware manual to find out which vector or vectors the device is capable of interrupting. To get a list of the vectors that are currently in use, enter:

   ./vectorsinuse

   A few drivers are written to allow vector sharing, but it is better to give each device a unique vector whenever possible. Associate the peripheral with an appropriate vector or vectors. Write down the vectors you choose.

9. Use the **configure** command to modify the system configuration files with the new driver information. All **configure** options are described in detail in the **configure**(ADM) manual page. The **configure** command has the following basic syntax and must be entered on a single line, without pressing **Return** until the entire command has been entered:

   ./configure -b -c -m *major_dev_number* **-s -v** *vector_list* **-a** *routines* \
   -l *interrupt_priority_level* **-h** *dev_name*

   These options have the following definitions and restrictions:

   -b Use if configuring a block device.

   -c Use if configuring a character device.

-m Should be followed by the major device number determined earlier.

-s When adding or deleting a streams module, use it with the **-h** option and instead of **-m**, **-b**, and **-c**. For a streams driver, use it with **-m** and **-c**.

-v Use only if the driver has an *intr* routine; should be followed by the list of vectors determined earlier.

-a Should be followed by the list of driver routine names determined by running **routines** and crossing out the extraneous entries.

-l Use only if the driver has an *spl* routine; should be followed by the interrupt priority level determined earlier.

-h Use only to give a device name that is different from the prefix of the driver routines or with a streams module when no prefix is specified; the driver's subdirectory in */etc/conf/pack.d* must use this same device name.

For example, to configure the serial I/O driver, use the command:

```
./configure -c -m 5 -v 3 4 -a sioopen sioread siowrite sioioctl \
siointr siopoll sioinit sio_tty -l 7
```

The ramdisk driver is a simpler example; you can configure it with:

```
./configure -b -m 31 -a ramopen ramclose ramstrategy ramprint
```

With the **-s** and **-h** options, you can configure a streams module:

```
./configure -a nmi_init -s -h nmi
```

10. Create a *device special file* in */dev* so programs can gain access to the newly installed device. The specific installation instructions supplied with the device will give the precise details of the name to be used for the special file and the other parameters associated with it. To create the device special file, use the

mknod command. Supply the name of the special file, the device type ("b" for block or "c" for character), the *major* device number, and the *minor* device number (indicating the unit, drive, or line number). For example, to create a special file for the serial I/O driver, enter:

/etc/mknod /dev/tty1a c 5 1

Here are some other examples of creating device special files:

/etc/mknod /dev/hcd0 b 1 0
/etc/mknod /dev/rhcd0 c 1 0
/etc/mknod /dev/hqp c 7 0

Note the UNIX convention for setting up disk device names. You can append a digit to the mnemonic to indicate the drive number. The "raw" device, or *character special* device, name has an "r" prefix.

11. Build a kernel containing the new drivers with the following command:

    ./link_unix

    Δ **sysadmsh** users select: System→Configure→Kernel→Rebuild

    Linking can take a while, so it is best to do this step only after you have installed all the drivers you want.

12. Boot the new kernel with the following command:

    /etc/shutdown

    Δ **sysadmsh** users select: System→Terminate

    A boot prompt appears. When you press **Return** to reboot the system, the new kernel is loaded and run.

If problems exist with the new kernel, reboot */unix.old*.

NOTE: If you attempt to select **System→Terminate** from within a window, it will fail. The easiest way to shut down the system is to log in as **root** and use the **shutdown** command.

Chapter 8
Using DOS and OS/2

Many users received the MS-DOS, or other closely compatible DOS, operating system with their computer. This chapter explains how you can still use DOS utilities, files, and applications after you install the UNIX system. You can even access DOS files and directories on your UNIX system, or mount DOS filesystems and access the files directly. The UNIX system provides this facility so that you do not need to throw away your investment in DOS software, or buy another computer just to run a UNIX system.

NOTE: This chapter is only concerned with accessing a DOS partition using facilities under the UNIX system. For information on accessing and executing DOS files under ODT-DOS, consult the ODT-DOS documentation.

Several programs make this coexistence possible. The **dos**(C) utilities allow access to DOS files on diskettes or on the DOS partition on the hard disk. These utilities are discussed later in this chapter. The utility that partitions the disk is called **fdisk**(ADM) and is available in DOS and UNIX versions. The next section explains how to use **fdisk** to create a DOS partition and a UNIX partition on the same hard disk. Another section discusses installing a UNIX partition on the hard disk along with DOS. There is also a section explaining various booting configurations, for users who mostly use the UNIX system and for users who mostly use DOS.

NOTE: You must have DOS 3.3 or earlier installed on your system. Extended DOS partitions are not supported.

OS/2 Coexistence

Although it may install successfully, OS/2 may not be bootable on your machine, regardless of whether a UNIX partition is present or not; we cannot guarantee that OS/2 will work with your UNIX system. Refer to your computer's hardware documentation to determine if your machine is supposed to run OS/2. If you wish to use OS/2 and/or DOS on the same disk with your UNIX system, you must load them in the following order:

1. DOS (partition must be 32MB or less)
2. UNIX software
3. OS/2

There are no OS/2 tools available (such as the DOS utilities described in this chapter). In addition, you must use **fdisk**(ADM) to switch to or from OS/2.

UNIX **fdisk**(ADM) displays an OS/2 partition as DOS.

Partitioning the Hard Disk Using fdisk

Each version of **fdisk** is documented in the respective operating system's manual. Unless otherwise noted, this chapter refers to the UNIX version of **fdisk**(ADM).

fdisk is interactive, and uses a menu to display your options. Here is the main **fdisk** menu:

```
  1. Display Partition Table
  2. Use Entire Disk For UNIX
  3. Use Rest of Disk for UNIX
  4. Create UNIX Partition
  5. Activate Partition
  6. Delete Partition

Enter your choice or 'q' to quit:
```

The **fdisk** utility allows you to set up separate areas (partitions) on your hard disk for your operating system. The hard disk is divided into *tracks*. The number of tracks depends upon the size of the hard disk.

A *partition* consists of a group of tracks. One hard disk may contain up to four partitions. Each partition can have a different operating system and associated directories and filesystems.

The **fdisk** command allows you to specify a disk partition as "active". This means that when you turn on (boot) your computer, the operating system installed in the active partition will start running. The UNIX partition must be active when you intend to use your UNIX system.

The **fdisk** command allows you to specify the number of tracks assigned to each partition. The number of available tracks will vary according to the size of your hard disk. Consult your *Release Notes* for the recommended UNIX partition size. The size of the UNIX partition also depends on the number of software packages you want to install. You can install the UNIX system in this space, and have the rest of the space for user files and other software packages. Refer to the **custom**(ADM) manual page for information on how to install and remove software.

The **fdisk** command allows you to specify where the partition begins and ends. **fdisk** will not allow you to construct overlapping partitions. You do not need to install your UNIX system in the first partition.

You should always start your DOS partition at the beginning of the disk, starting at cylinder 1, not cylinder 0. Because DOS writes the boot block on cylinder 0 very close to the end of the masterboot block, starting your DOS partition on cylinder 0 can cause the DOS partition to become inaccessible after installation.

If you install a UNIX partition on the same disk after DOS, start the UNIX partition at the beginning of the next cylinder on the disk. To find the beginning of the next cylinder, note the ending track number of your DOS partition and start the UNIX partition on the next track number that is a multiple of the number of heads on your hard disk. For example, if you have five heads on your hard disk and your DOS partition ends at track 103, start your UNIX partition at track 105.

When you are running your UNIX system, the device name of the UNIX partition is */dev/hd0a*. For more information about hard disk device names, see the **hd**(HW) page.

One option of **fdisk** tabulates the current state of the partitions (the Display Partition Table option). This option lists, for each partition, whether the partition is active, the first track, the last track, the number of tracks used, and the associated operating system. If you enter the Display Partition Table option and press **Return** to see the partition table, the result will be similar to this:

```
Current Hard Disk Drive:  /dev/hd00
```

Partition	Status	Type	Start	End	Size
1	Inactive	DOS	005	398	393
2	Active	UNIX	400	1219	819

```
Total disk size:  1229 tracks (9 tracks reserved for
masterboot and diagnostics).
```

Switching Operating Systems

There are three ways to switch to DOS once you have set up separate DOS and UNIX partitions:

- Enter **dos** at the boot prompt,
- Use a floppy diskette that contains the files necessary to boot the DOS operating system, or
- Use **fdisk** to change the current active partition.

We recommend that you use a boot floppy or enter **dos** at your boot prompt to boot the DOS operating system. Booting from a floppy or the boot prompt is generally easier, faster, and safer than constantly using **fdisk** to change active partitions.

When you use the boot prompt or a floppy to boot DOS, the UNIX partition remains active even though you have switched operating systems. When you use **fdisk**, the UNIX partition is inactive until you switch back to it.

To use the boot prompt method, enter:

dos

at the boot prompt:

```
SCO System V/386

Boot
:
```

To use a floppy diskette to boot DOS, follow this procedure:

1. Make sure all users are logged off the system.
2. Run **shutdown**(ADM) to shut down the UNIX system. This command makes sure all users know the system is being shut down, terminates all processes, then halts the system.
3. Once the UNIX system has shut down, insert the bootable DOS diskette into the primary (boot) drive.
4. Boot DOS.
5. To get back to the UNIX partition, remove any disks from the floppy drive(s) and press **Ctrl-Alt-Del**, or the reset key, or turn the computer off, then on. Since the UNIX partition is still active, your UNIX system boots.

Remember that if you have an active UNIX partition and boot DOS from a floppy you can transfer to C: to work with the DOS files.

The other way to change operating systems is to run **fdisk** and change the active partition from the UNIX partition to DOS. Then, after you shut down the system (see the previous steps) DOS boots from the hard disk. You do not need a bootable DOS floppy disk as long as DOS is loaded on the DOS partition of the hard disk.

To switch back to the UNIX partition, run **fdisk** under DOS and make the UNIX partition active. To reboot the UNIX partition, press **Ctrl-Alt-Del**, or the reset key, or turn the computer off, then on.

Because the UNIX partition must be active for it to operate, you cannot use a bootable floppy to boot the operating system. This second method is appropriate for an occasional change of the active operating system.

Table 8.1.

DOS Hard Disk Devices

XENIX device convention	UNIX device convention
/dev/hd0d	/dev/dsk/0sd
/dev/rhd0d	/dev/rdsk/0sd
/dev/hd1d	/dev/dsk/1sd
/dev/rhd1d	/dev/rdsk/1sd

The hard disk device names in Table 8.1 are similar to */dev/hd0a* (the active disk partition) in that the disk driver determines which partition is the DOS partition and uses that as *hd0d* and *hd1d*. (You can use the XENIX or UNIX device name conventions; they are equivalent.) This means that software that is running from the UNIX partition and using the DOS partition does not need to know which partition is DOS (the disk driver determines that).

Installing a UNIX Partition on a DOS System

If you wish to set up your UNIX system on a hard disk which previously contained only DOS, follow these steps:

1. Copy (back up) all the DOS files and directories on the hard disk onto floppies, or whatever backup media you wish to use.
2. Run **fdisk**, under DOS. If there is enough free space for a UNIX partition on your hard disk, (check your *Release Notes*) skip to Step 4. Otherwise, delete the DOS partition, then recreate it, leaving enough room on the disk for your UNIX distribution and any other software that you intend to install.
3. Return the DOS files from the backup media to the newly created DOS partition on the hard disk. Keep the backups in case there is an error of some kind, so you will not lose any data.
4. Turn off your computer.
5. Follow the installation procedure outlined in the *Installation Guide* to install your UNIX distribution.

   You will see a message warning that the contents of the hard disk will be destroyed. There is no cause for concern, because you have already backed up the DOS files and transferred them to the new DOS partition. The new partition being created will contain your UNIX system, and the installation process will only write information on the UNIX partition.

6. During the installation procedure, **fdisk** is invoked to partition the hard disk. Use **fdisk** to assign a sufficiently large UNIX partition.

7. Designate "UNIX" as the active operating system by choosing the "Activate Partition" option under **fdisk**.

8. Finish installing the UNIX distribution.

NOTE: UNIX **fdisk** displays DOS partitions as *DOS* while DOS **fdisk** displays UNIX partitions as *Other*.

You can only create DOS partitions using DOS **fdisk**, and only UNIX partitions using UNIX **fdisk**.

Be aware that DOS **fdisk** reports sizes in terms of cylinders, while UNIX **fdisk** reports sizes in terms of tracks. Check your hard disk manual for the number and size of cylinders on your hard disk.

Using a UNIX System and DOS with Two Hard Disks

Your computer always boots the operating system in the active partition on the first hard disk. The UNIX system must boot from the first hard disk. There are several ways to configure your system if you have two hard disks and want to boot DOS. Two ways are discussed here.

One configuration consists of designating the entire first disk as a UNIX partition. You then use a DOS boot floppy to start DOS and specify:

```
A>  A: C:
```

to switch to the DOS area on the second hard disk, where **C:** is the designation for the second hard disk. This strategy works for some versions of DOS. Early versions recognize only the first hard disk on the system.

NOTE: If you devote a hard disk for use with DOS, the disk must already be configured under DOS. See the "Adding Hard Disks" chapter of this Guide for details regarding hard disk configuration.

Another method is to maintain a small DOS partition on the first hard disk. The DOS partition is designated the active partition. In this configuration, the computer always boots DOS. This requires changing the active partition to boot the UNIX system from the hard disk.

If you use the entire second disk for DOS, you need only run **mkdev hd** to create device files for the second disk if you plan to use the UNIX DOS utilities (**doscp**, **dosls**, **doscat**, and so on). If you do not wish to use those utilities to access DOS files on the second hard disk, there is no need to run **mkdev hd**.

NOTE: Be sure to make a backup copy of your boot floppies if you use them to boot your secondary operating system.

Removing an Operating System from the Hard Disk

You may find that you no longer need one of the operating systems installed on your hard disk. If you want to delete an operating system, use the appropriate version of **fdisk**. To delete a UNIX partition, you must use the UNIX version of **fdisk**. To delete a DOS partition, use **fdisk** under DOS. Deleting the partition removes the contents of that partition and leaves unallocated space.

You can then reallocate that space by either adding another UNIX or DOS partition, or enlarging an existing partition. Enlarging a partition requires reinstalling the operating system and (for a UNIX partition) remaking the filesystem on the partition using **divvy**(ADM). Refer to the "Adding Hard Disks" chapter of this guide if you add a second UNIX partition and want to designate this partition as a mounted filesystem.

DOS Accessing Utilities

The DOS accessing utilities are discussed in detail in *Using ODT-OS* in the *User's Guide*. Note that you must have a bootable, although not active, DOS partition on the hard disk or a DOS floppy in order to use these UNIX commands. For example, you can only transfer a file from a UNIX partition on hard disk to a DOS floppy if either the DOS floppy is bootable or there is also a DOS partition on the hard disk.

You may also be able to use the UNIX **dd**(C) and **diskcp**(C) commands to copy and compare DOS floppies. The UNIX **dtype**(C) command tells you what type of floppies you have (various DOS and UNIX types).

Also, the file */etc/default/msdos* describes which DOS filesystems (e.g. A:, B:, C: ...) correspond to which UNIX devices.

The UNIX system does not record bad tracks in the DOS area of the hard disk. If a bad track develops in the DOS area, an operation such as **doscp** that attempts to access the affected area may fail. If such is the case, the message "Error on fixed disk" is displayed.

With smaller files, it may be possible to copy the files to another location under DOS and then access the copied version of each file.

NOTE: When trying to use the DOS utilities to access files on your DOS partition, you may see the error message "bad media byte." This message indicates that the DOS partition on the hard disk is not bootable. You can make your DOS partition bootable by first backing up the files on the DOS partition, booting DOS from the floppy, and formatting the DOS partition using the command:

format /s c:

You should now reinstall your DOS files.

File and Directory Arguments

The file and directory arguments for DOS files take the form:

device:name

where *device* is a UNIX pathname for the special device file containing the DOS diskette or DOS partition, and *name* is a pathname to a DOS file or directory. For example,

/dev/fd0:/john/memos

indicates that the file *memos* is in the directory */john*, and that both are in the device file */dev/fd0* (the UNIX special device file for the primary floppy drive). Arguments without *device:* are assumed to be UNIX files.

User Configurable Default File

For convenience, the user configurable default file */etc/default/msdos* can define DOS drive names that you can use in place of UNIX special device file pathnames. For example, you can include the following entries in the above file:

```
A=/dev/fd096ds15
B=/dev/fd048ds9
C=/dev/hd0d
D=/dev/hd1d
```

Once you have defined the variables, you can use the drive letter **A:** in place of the special device file */dev/fd0* (96ds15 by default) when referencing DOS files or directories. For example:

/dev/fd0:/john/memos

can be replaced with:

A:/john/memos

The drive letter **B:** refers to a low density (48ds9) primary floppy drive, and drive letters **C:** and **D:** refer to the DOS partition on a primary or secondary hard disk.

NOTE: If you get the message "cannot open */dev/hd0d*," or a similar message, check the user permissions on the special device file involved. As super-user, change the permissions with the **chmod** command. For example:

chmod 666 /dev/hd0d

gives full read and write permissions to all users for the special device file */dev/hd0d*, which is the DOS partition on the primary hard disk.

Mounting DOS Filesystems on a UNIX System

In addition to the DOS utilities provided with the Operating System to manipulate DOS files, it is also possible to mount a DOS filesystem and access its files freely while still operating from your UNIX system.

This means that DOS files can be edited or examined in place, without first copying them into the UNIX filesystem. The major restriction is that DOS files and applications cannot be executed under this arrangement; this requires use of VP/ix (if running under your UNIX system) or booting of the DOS partition. However, data files and text files can be examined, copied or edited.

Configuring Support for Mounted DOS Filesystems

In order to mount DOS filesystems, the support for these features must be present in the kernel. If it is not, you must first add this to your kernel with the **mkdev**(ADM) command. Make certain you are logged in as **root** and enter the following command:

mkdev dos

Δ **sysadmsh** users select: System→Configure→Kernel→DOS

This command adds the necessary functionality and prompts to relink the kernel. (If the link kit is not installed, you will be asked to install it.) After rebooting, you can mount DOS filesystems as described in the sections that follow.

How DOS Filesystems Are Accessed

The operating system deals with DOS filesystems by superimposing certain qualities of UNIX filesystems over the DOS filesystem without changing the actual files. UNIX filesystems are highly structured and operate in a multiuser environment. Thus they include many distinctions that have no meaning under DOS, including:

- File ownership
- Access permissions
- Special files (pipes, device files, etc.)
- Links

NOTE: Other applications/operating systems permit the mounting and access of DOS filesystems in this manner. However, most of them modify the DOS filesystem in some way to accomplish this. In the interests of portability, there are no proprietary modifications or extensions to the DOS filesystem. The ability to mount these filesystems is achieved purely through the facilities of the file system switch (FSS).

In order to make DOS files readily accessible, access permissions and file ownership are superimposed on the DOS filesystem when mounted.

Using the mount Command

The form for a DOS filesystem mount command is:

mount -r -f DOS /dev/hd*xy* */mountpoint*

where:

x	is the hard disk number
y	is the disk partition number
mountpoint	is the name of the directory in the **root** filesystem where the DOS filesystem is to be mounted.

The **-r** flag mounts the filesystem read-only, an optional precaution that will prevent damage to the DOS filesystem, which is not as robust as a UNIX filesystem.

When using **mount**, you must give the specific hard disk and partition numbers (as opposed to using wildcards).

Mounting a Floppy Disk

You can also mount DOS floppy disks, as in the following example using the 96tpi floppy mounted on */mnt*:

mount -r -f DOS /dev/fd096 /mnt

Repairing and Checking DOS Filesystems

The operating system includes a DOS version of the **fsck**(ADM) utility that works on DOS filesystems. This utility reconciles the DOS FAT (File Allocation Table) to the files contained on the filesystem. When **fsck** is invoked, it automatically detects the DOS filesystem and invokes the proper binary.

Who Can Access the Mounted DOS Filesystem

Only **root** can mount a filesystem. Access by users is governed by the permissions and ownership that **root** places on the DOS filesystem. Because of the limitations discussed earlier, DOS does not recognize permissions or ownership. When mounted on a UNIX system, the DOS files behave as follows:

- The permissions and ownership of the filesystem are governed by the mount point. For example, if **root** creates a mount point */x* with permissions of 777, all users can read or write the contents of the filesystem. If the mount point is owned by **root**, all files within the DOS filesystem and any created by other users are all owned by **root**.

- The permissions for regular files will be either 0777 for readable/writable files or 0555 for read only files. This preserves the consistency of the DOS filesystem. If a user can access the filesystem, the user will be limited by the permissions available under the DOS directory structure. This permission is read-only or read-write. When a file is created, the permissions are based on the **umask** of the creator. For example, assume the user's **umask** is 022, which generates files with permissions of 777. Here are further examples.

  Example 1: Creating a file. The permissions are based on the **umask** owner section. A umask of 022 will provide a file of 777 on the DOS partition. Because the owner has not masked off the write bit for themselves.

  Example 2: Examining a file already on the DOS partition. The permission you see is the logical AND of the UNIX mountpoint permission and the DOS file permission. So, a UNIX mountpoint of 750 and a DOS file permission of 555 will get you 550 for the permissions. This has nothing to do with the **umask**.

- There can only be 1 link for each file under the DOS filesystem. "." and ".." are a special case under this arrangement and are not links as they are on a UNIX system.

- On UNIX systems, features such as locking govern how, under certain programs and applications, a file is accessed simultaneously by different users. These features operate identically on a mounted DOS filesystem. Two users can edit the same file and write to it as permitted by the locking mechanism used.

Appearance of DOS Files

Because no attempt is made to change the nature of DOS files, the carriage return character (^M) will be visible when editing a DOS file on a UNIX system. (UNIX systems use only a newline, while DOS uses a carriage return and a newline.) The **dtox**(C) and **xtod**(C) commands are the easiest way to switch the end-of-line format. **dtox** is used to change DOS format to UNIX format, and **xtod** vice-versa. These tools are described in more detail in *Using ODT-OS* in the *User's Guide.*

Restrictions

There are additional logical restrictions that must be observed.

File Names

The rules for file names and their conversion follows the guidelines found in the **dos**(C) manual page. In addition, the standard DOS restrictions on illegal characters apply. However, wildcards can be used just as they can with a UNIX system.

Modification Times

When accessed from the UNIX partition, the creation, modification, and access times of DOS files are always identical and use GMT, or Greenwich Mean Time. (This is because UNIX uses GMT internally and converts it for the user.) This means that files created in the DOS filesystem while under DOS or UNIX will not have consistent times across the operating systems.

UNIX Backup Utilities

The **backup**(ADM) and **xbackup**(ADM) utilities cannot be used to make backups of a mounted DOS filesystem. DOS utilities and other copy programs like **tar**(C) will work as expected.

For more information, including more technical aspects of DOS usage, refer to **dos**(C).

Chapter 9

Administering User Accounts

User accounts help the system administrator keep track of the people using the system and control their access to system resources. Each account has a unique "login name" and "password" with which the user enters the system, and a "home directory" where the user works. In addition, the system has certain defaults that define how long a user password should last, whether users are allowed to choose their own passwords, and how many unsuccessful login attempts should be allowed before locking a user out.

It is the system administrator's job to create accounts for all users on the system, and maintain these accounts by changing user passwords, login groups, and user IDs when necessary.

This chapter explains the following functions:

Account Management	Add, alter, and remove ("retire") user accounts, plus creation of user groups
Default Account Configuration	Configure system default login and password parameters
Activity Report Generation	Produce reports on user logins, terminal usage, and password status

It is important to examine the default account restrictions soon after creating user accounts. These are summarized in "Default Account Configuration." You should determine if these defaults are appropriate to the needs of your system.

NOTE: Under no circumstances should you edit /etc/passwd with a text editor. On other UNIX systems, this is a common, but untrusted way of adding users. This will cause error messages to be displayed and could cause the system not to accept further logins. Use the **sysadmsh** Accounts selection to modify or add user accounts. The */etc/passwd* database has been expanded into an adjunct Protected Password database, which stores the encrypted version of the password and other security parameters about each user.

Account Management

This section explains how to create and manage user accounts.

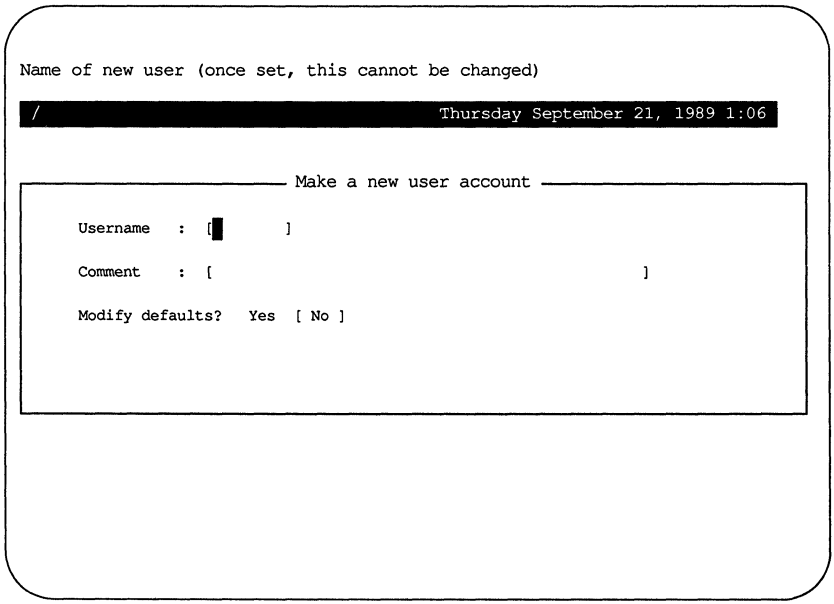
Adding a User

You can add a user account to the system with the **sysadmsh** program. The program creates a new entry in the accounts database. The database contains information about the new user (such as login name and initial password) that the system uses to let the user log in and begin work. **sysadmsh** also creates a home directory for the user, a mailbox for use with the **mail** command, and an initialization file (for example, *.profile* for the Bourne shell or *.login* for C-shell) containing UNIX commands that are executed when the user logs in.

To create a user account, return to the desktop and double-click on the Sysadmsh icon to bring up the main **sysadmsh**(ADM) menu. Then make the following selection:

Accounts→User→Create

The following screen is displayed:



Follow these steps steps to add a user:

1. Fill in the username and, if desired, comment fields.

2. If you wish to alter the defaults, select "Yes" and define the fields as shown in "Altering User Defaults." Fill in each field as necessary; pressing **F3** allows you to choose from point-and-pick lists. When you press **Return**, the field is filled in with the value you selected.

3. When you exit the form, a window pops up to confirm your additions. If confirmed, a series of creation messages are displayed that look like this:

```
Created home directory: pathname
Created shell file: filename
Greetings mail sent to user: name
```

   This indicates that all the necessary files and directories were created. (This default information is taken from */usr/lib/mkuser*.)

4. The final step is initial password creation. **sysadmsh** prompts you as to whether an initial password should be created. If no password is assigned, no-one may log into the account. Select "yes" and follow the password assignment procedure:

```
Last successful password change for user: date
Last unsuccessful password change for user: NEVER

                    Choose password

You can choose whether you pick your own password,
or have the system create one for you.

        1. Pick your own password
        2. Pronounceable password will be generated for you

Enter choice (default is 1):
```

5. If you select **1**, you are prompted to enter the password, twice:

```
Password:
Re-enter password:
```

   Note that the password is not displayed on the screen as you enter it.

6. If you select **2**, the following is displayed:

```
Generating random pronounceable password for user.
The password, along with the hyphenated version, is shown.
Hit <RETURN> or <ENTER> until you like the choice.
When you have chosen the password you want, type it in.
Note: Type your interrupt character or 'quit' to abort at any time.

Password:xxxxxxxx  Hyphenation:xx-xx-xx Enter password:
```

   The generated password is displayed with a hyphenated version. The hyphenation separates the password into pronounceable syllables and is designed to help you commit the password to memory.

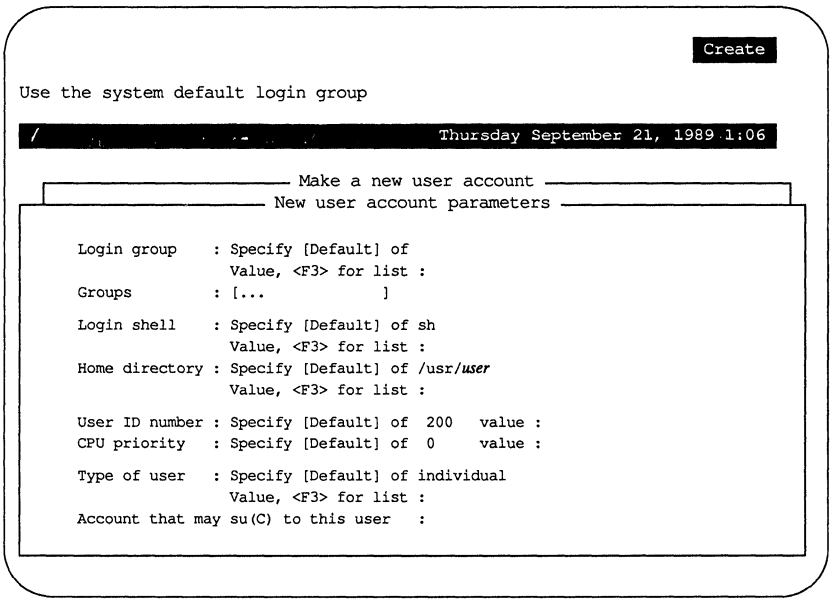
7. Give the new password to the user, advising them to change it immediately after logging in.

The new account is usable and will be maintained according to the default security parameters unless you have set specific values for the user.

Altering the User Defaults

For most users, simply select **Identity** and define the information shown below. The other categories (Audit, Logins, Password, Authorizations) are subject to the system defaults. You need not select the other categories unless you wish to define specific capabilities or limitations for that user.

The "Defaults" form looks like this:



The cursor is initially positioned on the "Login group" field. Some of the fields displayed can only be modified at creation time - in the *Modify* mode, these fields are informational only; their values cannot be changed.

Login group

Group associated with this account when the user logs in. This field can be changed, but must not be empty. It will become the group field for this user in */etc/passwd*. Pressing function key **F3** provides a point-and-pick list of all currently existing groups. Note however, that doing so will *destroy* the previous contents of this field, even if no group was picked from the point-list.

If the user is not currently a member of the chosen group, the user will be added and the screen redrawn so that the group appears in the "Groups" field.

The user will *not* be removed from the old (previous) login group; to do that the administrator must delete the applicable group from the "Groups" field. When the first group name is selected, another window opens up in the middle of the screen; this window remains open so that multiple groups can be entered. For each group that does not exist, an alert box is displayed prompting you to create the group. When you are finished entering groups, press **Return** to move on.

Groups
The list of groups this user is a member of; it is not a fill-in field. It displays the additional groups entered in the window opened by the "Login Group" field.

Login shell
This is the shell the user will use. (The default is defined in */etc/default/authsh*.) If a full pathname is supplied (as in "/bin/sh"), the shell described by that pathname is simply used as the user's login shell. However, if the shell specified is not a pathname (as in "sh") it is assumed to be the name of a "pre-defined shell", a shell defined in a subdirectory of */usr/lib/mkuser*. Choosing a pre-defined shell causes appropriate shell-related files (example: *.profile* for "sh") to be copied into the user's home directory when the account is created. To configure an account so that Open Desktop is automatically brought up when the user logs in, select **odtsh**.

Home directory
This field defines where the user's files will reside. Press **Return** to get the default location.

User ID
The user ID number. Once set, a user's identity cannot be changed as this would obscure the audit trail.

CPU Priority
CPU scheduling: zero is the default; the lower the value, the higher the priority. This value can be negative. This corresponds to the **nice**(C) value; see the manual page for more details. This field can be changed but must not be empty. The priority can be changed to penalize users running programs that use excessive CPU time.

Type of user
In most cases, this is "individual" or "pseudo-user". By default this field assumes an individual. Individual users are real people with names. Pseudo-users are anonymous accounts like **mmdf**. Each pseudo-user account must have an "accountable user," who is considered responsible for that account.

Account that may su(C) to this user
User responsible for this account. This field can be changed if and only if the user is *not* an *individual*. For *individual* users this field is empty, but for non-individuals it must not be empty. It must contain the username of an individual account (not retired). For example, the **root** account must have the name of a user who is responsible for the account. This ensures that no account is anonymous; every account must be traceable to a real person. Press **F3** for a point-and-pick list of all individual users on the system.

Adding Administrative Users

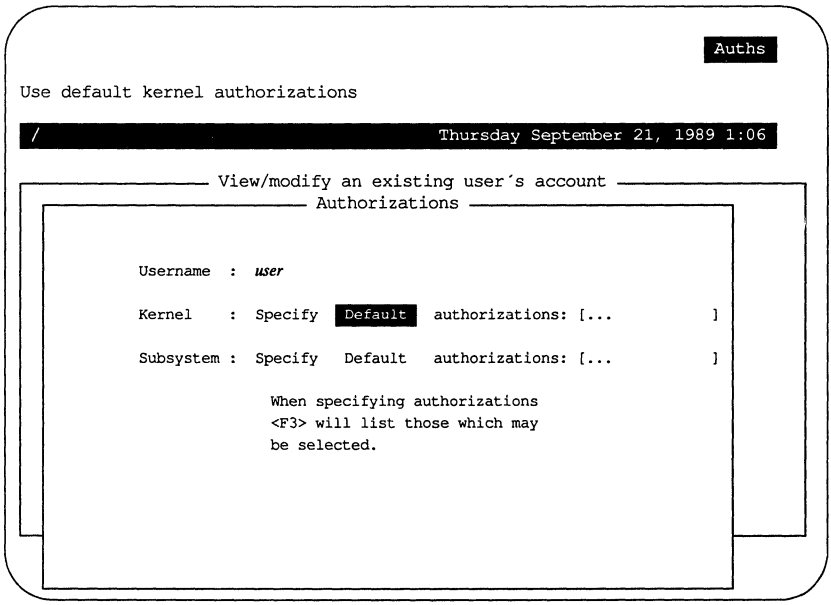
In addition to the standard **Identity** information, users who will act as administrators for printers, accounts, etc., can be assigned the responsibility by selecting **Privileges.** Subsystems are discussed in "Changing Default Authorizations," and assigning authorizations is discussed in the next section.

Altering/Assigning User Authorizations

Subsystem authorizations are discussed earlier in "Changing Default Authorizations" and in greater detail in the "Maintaining System Security" chapter. Authorizations are intended to be assigned only to users entrusted with administration of a subsystem. To assign a new authorization to a user, return to the desktop and double-click on the Sysadmsh icon to bring up the main **sysadmsh**(ADM) menu. Then make the following selection:

Accounts→User→Examine:Privileges

The following form is displayed:



You can select "Specify" and press **F3** to open a window that lists the available authorizations.

NOTE: If you switch from defaults to specified, the default values are eliminated for that user; only those authorizations you specify are in effect.

Removing (Retiring) a User Account

In the strictest sense, a user is never removed from the system. A user ID, once assigned, is never re-used. Instead, a user account is "retired," or removed from service. To retire a user account, return to the desktop and double-click on the Sysadmsh icon to bring up the main **sysadmsh**(ADM) menu. Then make the following selection:

Accounts→User→Retire

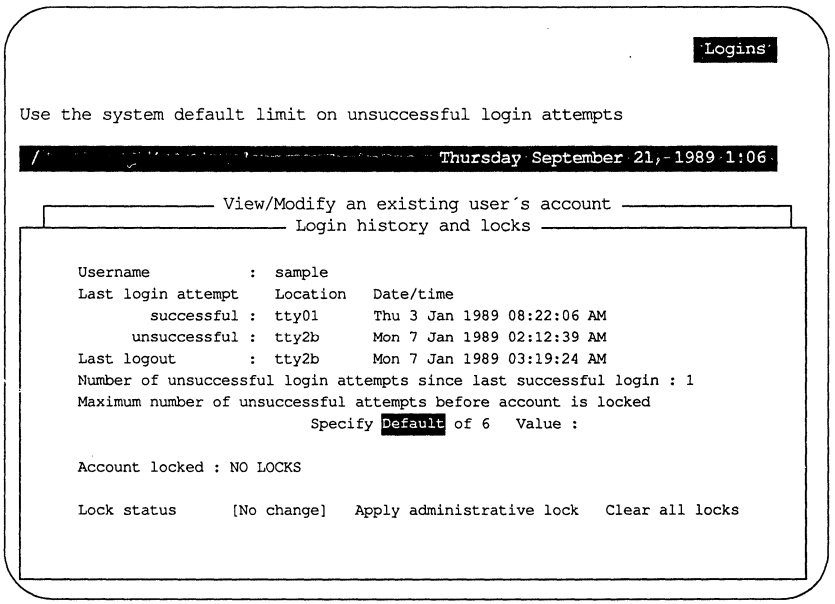
NOTE: A retired account can never be reactivated. Retirement is permanent.

Locking/Unlocking a User Account

The system administrator can lock an account to prevent its use. In addition, an account will be locked automatically if the login parameters have been exceeded (see "Default Account Configuration"). Once a user or terminal is locked, only the administrator can unlock the user account or terminal. To lock or unlock an account, return to the desktop and double-click on the Sysadmsh icon to bring up the main **sysadmsh**(ADM) menu. Then make the following selection:

Accounts→User→Examine:Logins

A form similar to the following is displayed:



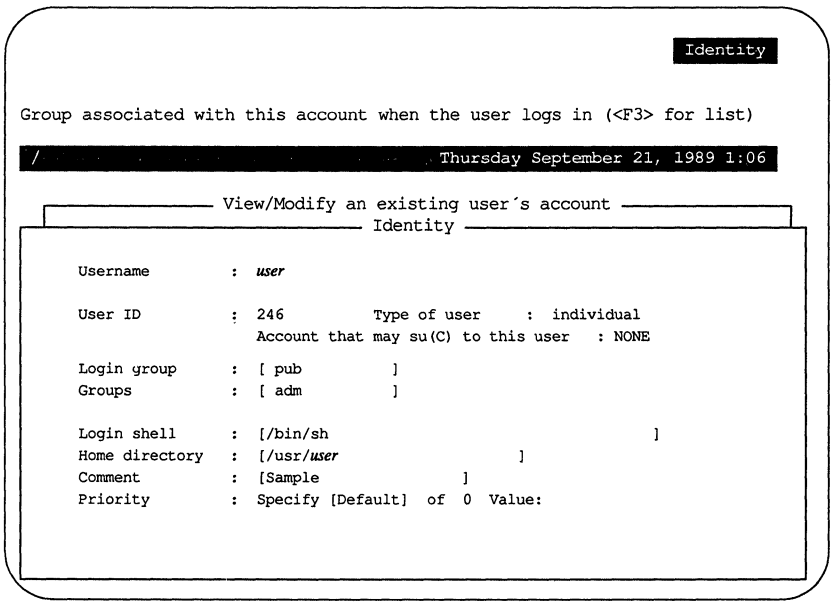
Move down to the "Lock status" field and toggle it to *Apply administrative lock* or *Clear all locks* as desired.

Changing a User Group

To change a user's login group, return to the desktop and double-click on the Sysadmsh icon to bring up the main **sysadmsh**(ADM) menu. Then make the following selection:

> **Accounts→User→Examine:Identity**
> The colon indicates that you must fill in a field (in this case the user name) before choosing the *Identity* selection.

A form similar to the following is displayed:



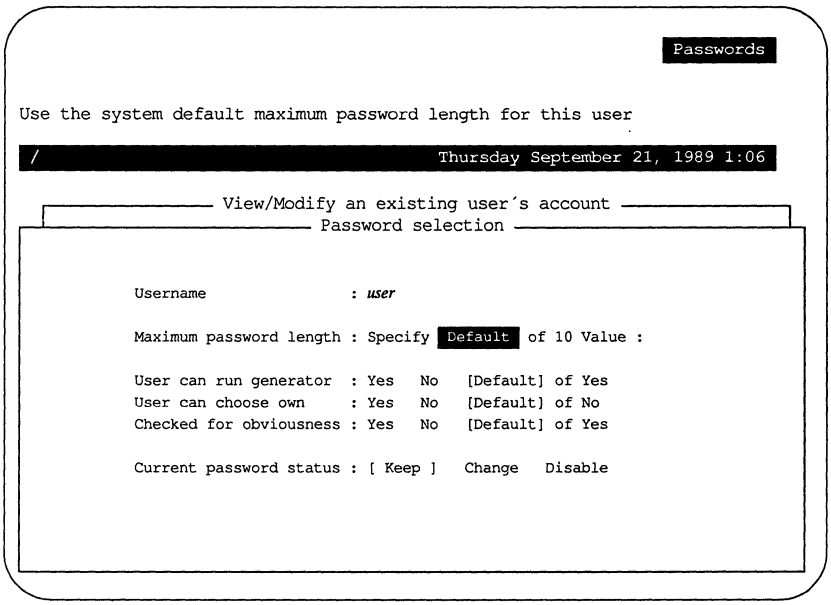
Modify the Login Group field as desired.

Changing a User Password or Password Parameters

An administrator can change a user's password at any time. Password generation parameters can also be changed on an individual basis, just as they can be system-wide. This governs how a user's password is changed. To do this, return to the desktop and double-click on the Sysadmsh icon to bring up the main **sysadmsh**(ADM) menu. Then make the following selection:

Accounts→User→Examine:Password

The following form is displayed:



Complete descriptions of the password parameters are found in "Changing Default Password Restrictions."

You can change the user's password by selecting "Change" from the Current password status. This invokes the password change procedure described at the end of "Adding a New User."

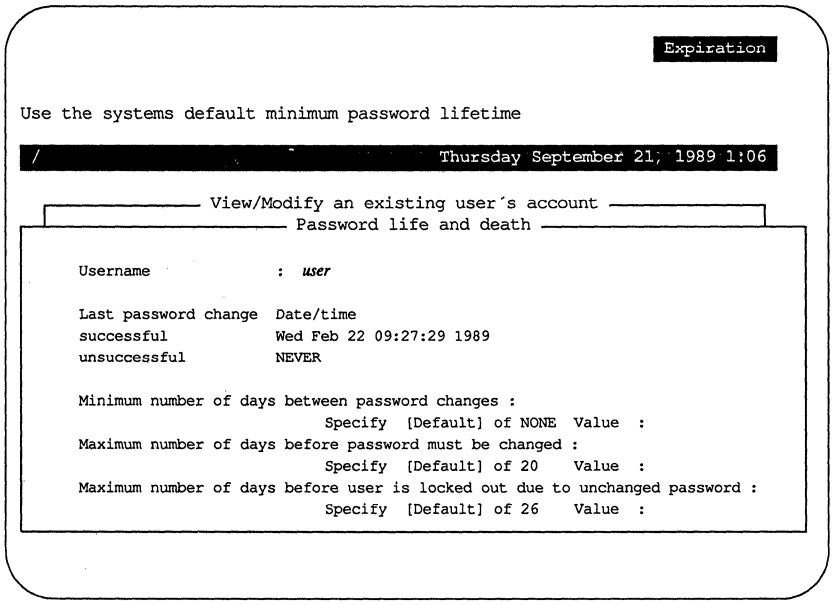
You can also "Disable" the password, which effectively locks the user out.

Altering User Password Expiration Parameters

It is sometimes useful to define expiration parameters for a user that differ from the system defaults. To do this, return to the desktop and double-click on the Sysadmsh icon to bring up the main **sysadmsh**(ADM) menu. Then make the following selection:

Accounts→User→Examine:Expiration

A form similar to the following is displayed:



The user parameter descriptions are similar to the system-wide parameters described in "Changing Default Password Restrictions." The descriptions differ, but the parameters are the same. The password restrictions are as follows:

Minimum number of days between password changes
The number of days a user must wait before they can change their password.

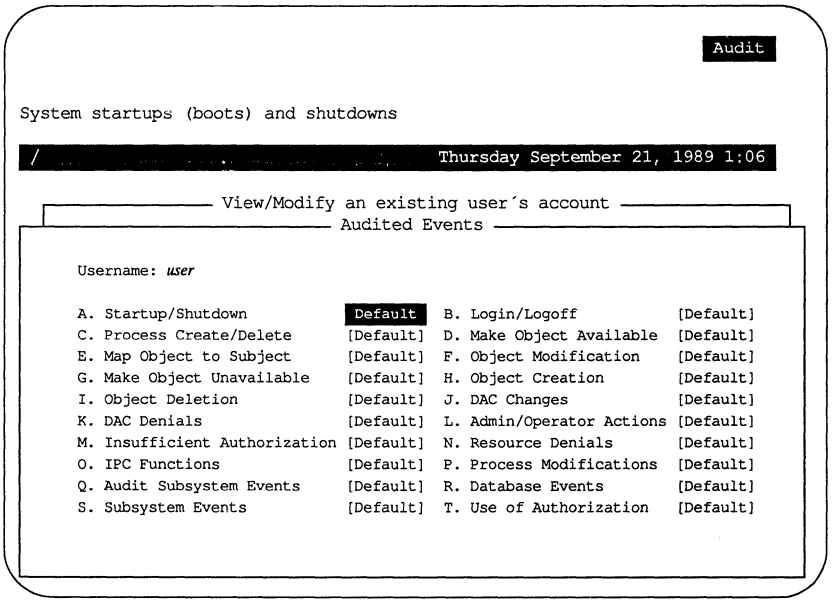
Maximum number of days before password must be changed
This defines the length of time a given password is valid.

Maximum number of days before user is locked out due to unchanged password
This defines the interval between last password change and when the password dies.

Defining/Changing User Audit Parameters

You can define audit parameters for individual users just as with the system-wide parameters. Any settings defined for a user override the system defaults. To define or change audit parameters, return to the desktop and double-click on the Sysadmsh icon to bring up the main **sysadmsh**(ADM) menu. Then make the following selection:

Accounts→User→Examine:Audit



A detailed description of audit events is found in the "Using the Audit Subsystem" section of the "Maintaining System Security" chapter in this guide.

There are three possible settings: "Default", "Always", and "Never". You can press **F3** to select from a list of these settings or fill them in manually. Abbreviations are recognized (for example, "n", "nev", and "N" all mean NEVER). To execute the form, press **Ctrl-x**. (If you fill in the last field on a form, it is automatically executed.)

Adding/Changing Groups

To add a group, simply enter a new group name while creating or altering a user account. You will be prompted that the group does not exist and asks you to confirm that you wish to create the new group.

Changing the Maximum Number of Groups

By default, the maximum number of groups that a user can be associated with is 8. This number is controlled by the NGROUPS tunable kernel parameter. This value can be changed by invoking the **sysadmsh** selection **System→Configure→Kernel→Parameters** and selecting category 3: "Files, Inodes and Filesystems" and changing the value of NGROUP. The kernel must then be relinked and booted for the new value to take effect. Use the **sysadmsh System→Configure→Kernel→Rebuild** selection to relink the kernel.

Default Account Configuration

This section explains how to alter the system security defaults, which include default password schemes, subsystem authorizations and number of login attempts permitted for users.

Your operating system distribution comes preconfigured with a set of defaults that define the security scheme used for accounts. Table 9.1 includes these defaults, including the "Relaxed" values (consistent with other, less secure UNIX systems) that can be selected as described in "Selecting the Relaxed Security Defaults":

Table 9.1.
System Default Security Parameters

Security Parameters	Relaxed	C2
Passwords		
Minimum days between changes	0	14
Expiration time (days)	0	42
Lifetime (days)	0	365
Maximum password length	8	10
User can run generator?	yes	yes
User can choose own?	yes	yes
Passwords checked for obviousness?	no	no
Single user password required?	yes	yes
Logins		
Maximum unsuccessful attempts	99	5
Delay between login tries	0	2
Audit Event Types		A, B, F, H, I, J, K, L, M, N, Q, R, S, T†
Authorizations		
Subsystem	queryspace, printerstat, printqueue, mem, terminal	queryspace, printerstat printqueue
Kernel	execsuid, chmodsugid, chown, nopromain	execsuid, chown, nopromain
Default umask*	022	077

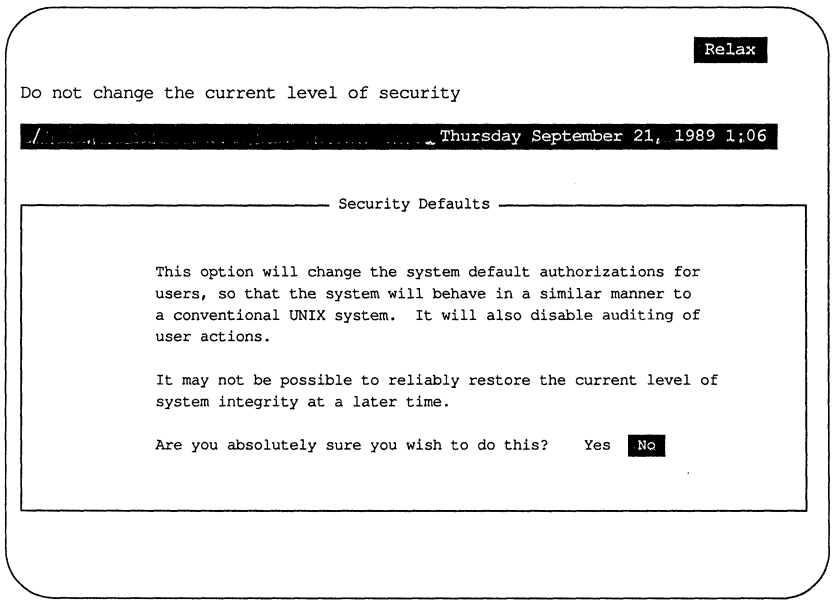
† Audit event types are described in the "Maintaining System Security" chapter in this Guide.

* These are located in */etc/profile* and */etc/cshrc*. A **umask** of 077 results in the creation of files that are readable only by the owner. When "relaxed" is selected, the **umask** value is not changed if the existing value has already been altered.

You were given the choice of security defaults at installation time. If you chose the "C2" defaults, it is possible to change them. Should you wish to "downgrade" your system to function in a way similar to other UNIX systems, return to the desktop and double-click on the Sysadmsh icon to bring up the main **sysadmsh**(ADM) menu. Then make the following selection:

System→Configure→Security→Relax

This selects an alternate set of defaults that define a security policy consistent with less trusted UNIX systems (see Table 9.1). When you make this selection, the following warning is displayed:



If you made any changes to your security defaults, you are given the following additional warning:

```
The default file has changed since installation
        Previous changes will be lost

Press <Return> to continue, or <Escape> to abort
```

Selecting the C2 Security Defaults

After having selected the "Relaxed" defaults, it is possible to restore the C2-level defaults, although this does not mean that your system automatically conforms to the requirements of a C2 system. (By definition, a C2 system must adhere to the requirements from initial installation.) To restore the C2 defaults, use the chart in Table 9-1.

Changing Security Parameters Dynamically

To access the system-wide account parameters, return to the desktop and double-click on the Sysadmsh icon to bring up the main **sysadmsh**(ADM) menu. Then make the following selection:

Accounts→Defaults

The following account parameters are displayed:

- Authorizations
- Password
- Logins

The system-wide security parameters control the way that users log in and, once they establish a session, the terminal and authorization environment that the system presents to them. Each parameter that you can change from the **sysadmsh** interface is discussed here. Other parameters used by that affect system operation are addressed later.

You should use the system-wide functions to define your own default system behavior. Then use the user-specific functions to adjust that behavior for any user having different requirements. As you might expect, the user-specific entries override the system defaults for any given user.

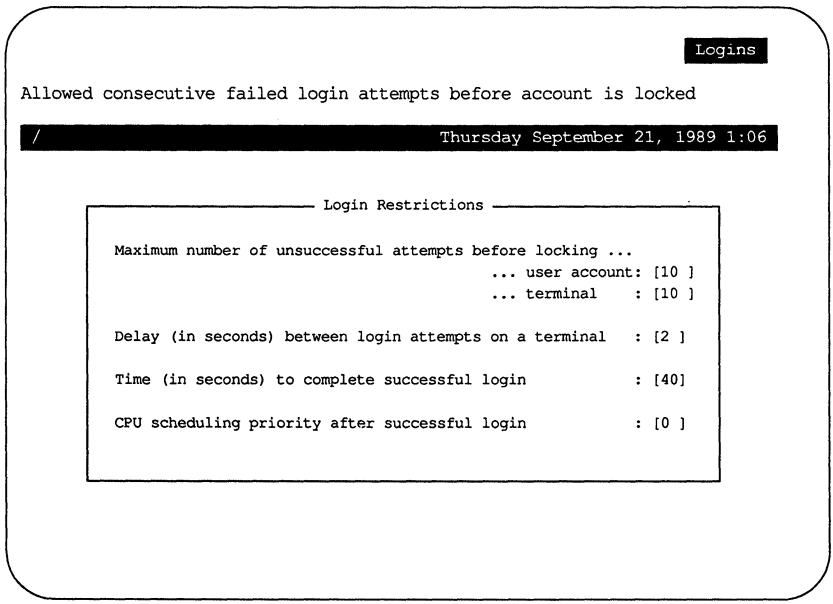
Changing Default Login Restrictions

Most of the parameters that can be set on the system defaults deal with the way the system creates a login session. These include login particulars, and the way that passwords are generated and enforced. The login parameters enforce the account and terminal locking features. When users logs in, they must give a login name and password. In addition, the user has a limited number of tries to log in. There is a limit on the number of times an unsuccessful login attempt can occur before either the account or the terminal are *locked.* If either count is exceeded, the user or the terminal is locked against future login. This feature guards against penetration attempts by restricting the number of times a malicious user (or computer programmed by a malicious user) can try to break into the system.

To access the login restriction parameters, return to the desktop and double-click on the Sysadmsh icon to bring up the main **sysadmsh**(ADM) menu. Then make the following selection:

Accounts→Defaults→Logins

The following form is displayed:



The parameters are described as follows:

Maximum number of unsuccessful attempts before locking
This is the system default number of unsuccessful attempts allowed for users and terminals. If a particular user or terminal needs either a more restrictive or more permissive count, the user's account can be modified or the terminal's configuration (see "Terminal Login Management") can be changed to override the system default.

Delay (in seconds) between login attempts on a terminal
This parameter controls the amount of time that must pass between unsuccessful login attempts. To further reduce the possibility of penetration, the system can delay between login attempts to increase the amount of time it takes to repeatedly try to log into the system. You can increase this parameter to control the cycle time of the *login:* prompt. By combining this parameter with the user and terminal unsuccessful attempt parameters, you can frustrate attempts to repeatedly try passwords on a certain (or a combination of) terminal lines.

Time (in seconds) to complete successful login
This parameter determines how much time a user has to enter their name and password before the login attempt is terminated.

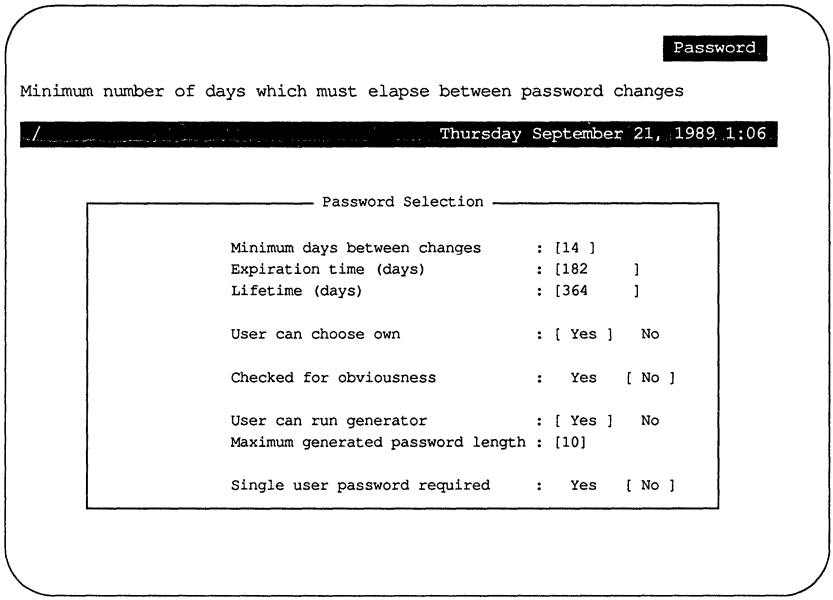
CPU scheduling priority after successful login
This sets the **nice**(C) value associated with this user's processes.

Changing Default Password Restrictions

To access the password restriction parameters, return to the desktop and double-click on the Sysadmsh icon to bring up the main **sysadmsh**(ADM) menu. Then make the following selection:

Accounts→Defaults→Password

The following screen is displayed:



Given that you can control the number of attempts an intruder can try to guess a password, it remains to control the complexity of the password itself. The types of password checking the system does is controlled by several parameters you set on this screen. The parameters control the time that a password is valid, and the procedures for changing the password once it becomes invalid. A password is valid until it *expires* or *dies*. An expired password can be changed by whoever is authorized to change passwords for the account. A password expires when its expiration time is reached. The expiration time can be set from the administrator interface on a system-wide or a per-user basis, and is expressed in a number of days from the time that the password was last changed. A dead password causes the user account to be locked. Only the administrator can unlock the user's account, which is then treated as an account with an expired password. The password must still be changed before the user can log in again.

To discourage users from changing their passwords when they expire and then immediately changing them back to one they remember, the system also stores a *minimum* time between password changes. A user's password cannot be changed until the minimum time has been exceeded. This parameter may also be set on a per-user or system-wide basis.

The following parameters define the password restrictions:

Minimum days between password changes
: The number of days a user must wait between password changes.

Expiration time (days)
This defines the length of time a given password is valid.

Lifetime (days)
This defines the interval between last password change and when the password dies.

Maximum password length
The maximum length of a password. The system maximum is 80 characters.

User can run generator
This parameter is used to enable users to run the password generator. Note that this does not allow users to choose a password, merely generate a new random password.

User can choose own
This parameter determines whether or not users can choose their own passwords. A "trusted" system requires that the system must generate passwords automatically for users. This guards against users picking "obvious" passwords that a knowledgeable intruder could guess given some personal facts about the user. Other UNIX systems, however, allows users to pick their passwords. If this parameter is set to **yes**, then rules consistent with less-trusted UNIX systems are in effect, allowing users to pick their passwords. If that parameter is set to **no**, the system must generate passwords for that user, according to the random password generation procedures.

Checked for obviousness
This parameter tells whether the system should run triviality checks on the resulting password. These checks assure that the password does not appear in the on-line dictionary, along with the other checks described in **goodpw**(ADM). Setting this parameter to **yes** assures that some penetrations based on trying all real words fails, but this can more effectively be controlled through the limits on user and terminal logins. The triviality checks increase the time required to change a password substantially. All three of these parameters can be overridden through the user-specific parameters.

Single User password required
This governs whether a password is required to bring the system up in single-user (maintenance) mode.

When an account is locked by the system, only **root** or the Accounts Administrator can unlock it. The password must be changed at that time. You can override these parameters for any user by setting up that user's parameters as described in "Adding a User."

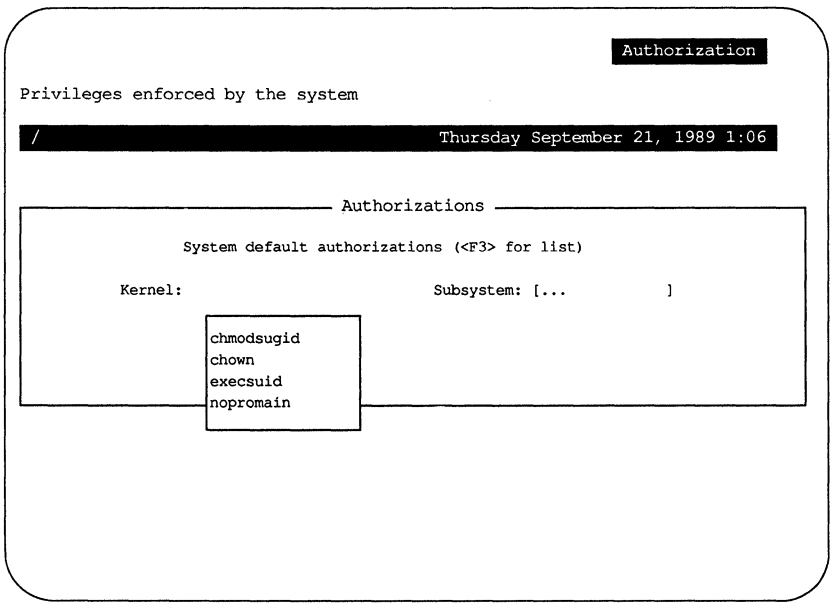
Changing Default Authorizations

The operating system defines two types of authorizations: kernel authorizations and subsystem authorizations. *Subsystem authorizations* are associated with users and allow the user to execute trusted utilities. *Kernel authorizations* are associated with processes and allow a process to perform certain actions if the process has the requisite authorization. Each user session has a set of kernel authorizations and a set of subsystem authorizations.

To access the authorization parameters, return to the desktop and double-click on the Sysadmsh icon to bring up the main **sysadmsh**(ADM) menu. Then make the following selection:

Accounts→Defaults→Authorizations

The following screen is displayed:



Use the **F3** key to get a pop-up window that lists each set of authorizations. The descriptions follow.

Table 9.2.

Subsystem Authorizations

Authorization	Subsystem	Powers
mem	Memory	Access to "private" system data; listing all processes on the system
terminal	Terminal	Unrestricted use of the **write**(C) command
lp	Line Printer	Printer administration
backup	Backups	Performing backups
auth	Accounts	Accounts Administrator: adding users, changing passwords, etc.
audit	Audit	Audit Administrator: run system audits and generate reports
cron	Job Scheduling	Control use of **cron**(C), **at**(C), and **batch**(C) commands
sysadmin	System Integrity	Ability to run **integrity**(ADM) program

The subsystem authorizations determine the *administrator roles* that a user may assume by running the trusted utilities. For a general system user, no subsystem authorizations are allowed. The administrative staff are granted subsystem authorizations based on their responsibilities; that is, the Accounts Administrator is given *auth* authorization and the Printer Administrator is given *lp* authorization.

In the System Defaults database, a default set of command authorizations is given to all users that do not have command authorizations in their user-specific account information. In the case of the C2 security defaults, the subsystem authorizations in the System Defaults database are empty and the user-specific entries are set based on the administrative roles, if any, of that user.

The **sysadmin** subsystem authorization controls the power to run the **integrity**(ADM) program, which checks the permissions of files listed in the File Control database. (For more information, refer to the **integrity**(ADM) manual page and "System Integrity Checking" in the "Maintaining System Security" chapter of this guide.)

Table 9.3.

Secondary Authorizations

Secondary Authorization	Subsystem	Description
queryspace	backup	Use **df** command to query disk space
printqueue	lp	View all jobs in queue using **lpstat**
printerstat	lp	Use printer enable/disable commands
su	auth	Grant a user access to the **root** (super-user) account. (Still requires root password.)

These secondary authorizations allow limited access by users to resources that would otherwise be tightly controlled (for example, without the *printqueue* authorization, a user would only be able to see their own jobs when they use the **lpstat** command). They provide behavior that is more consistent with other UNIX operating systems. With the "Relaxed" defaults, secondary authorizations are granted by default. If you are using the "Relaxed" defaults and you wish to prevent users from having these authorizations, you must redefine the defaults.

NOTE: When the primary authorization for a subsystem is granted, the secondary authorizations for that subsystem are also granted. (For example, the **lp** authorization carries the **printqueue** and **printerstat** authorizations.)

The Super-user Versus Authorized Administrators

Most of the powers normally exercised by the superuser on a less secure system have been assigned in to the protected subsystems discussed earlier. However, some functions still need to be done by user **root**. It is important to keep this in mind when assigning authorizations. Assignment of the **su** authorization allows an administrative user to **su**(C) to the super-user account.

Super-user powers are required to perform the following tasks:

1. Software Installation
2. Disk partitioning and file system maintenance
3. File restoration, recovery, and permission setting
4. System shutdown
5. Troubleshooting

Kernel Authorizations

The kernel authorizations govern the power that users have to execute specific operating system services. For example, the ability to change ownership of a file is governed by the **chown** authorization. The default kernel authorizations are used whenever a user's kernel authorizations are unspecified. Thus, users that need more authorization can have user-specific entries that grant them those authorizations, while normal users can have their authorizations set to the *default* contained in the System Defaults database.

Table 9.4.

Kernel Authorizations

Authorization	Action
configaudit	Configure audit subsystem parameters
writeaudit	Write audit records to the audit trail
execsuid	Ability to run SUID programs
chmodsugid	Ability to set the SUID and SGID bit on files
chown	Ability to change the owner of an object
suspendaudit	Suspend operating system auditing of the process.
nopromain	Access as user outside the promain directory

The restrictions provided by these authorizations are complex; they are configured by default to function under the requirements of the C2 level of trust. The audit parameters apply specifically to audit operations and should not be assigned to users; these parameters are explained in the **subsystem**(M) manual page.

Kernel Authorizations and Administrative Users

You must assign certain kernel authorizations along with subsystem authorizations. Although most of these are already assigned by default, they are listed in Table 9.5 in case you modify the defaults. One exception is the Audit subsystem, which requires the addition of the *configaudit* and *suspendaudit* authorizations. These authorizations should never be assigned by default, or to ordinary users. Another exception is the **sysadmin** authorization, which requires the **chmodsugid** kernel authorization, though it is simpler to run the **integrity**(ADM) program as **root**.

Table 9.5.

Subsystem Kernel Authorization Requirements

Subsystem Authorization	Required Kernel Authorization
audit	configaudit, suspendaudit, execsuid
auth	chown, execsuid
backup	execsuid
lp	chown
cron	execsuid, chown, chmodsugid
sysadmin	execsuid, chmodsugid, chown

Locking/Unlocking a Terminal

To lock and unlock a terminal, respectively, use the following **sysadmsh** selections:

Accounts→Terminals→Lock

Accounts→Terminals→Unlock

When the prompt appears for the terminal, enter the name, for example: tty01. When a terminal is locked, the following message is displayed when an attempt is made to log in:

```
Terminal is disabled -- see Authentication Administrator
```

Activity Report Generation

It is possible to create reports on the status of three important aspects of system operation:

Passwords Report on accounts by password status

Terminal Report on access by terminal status

Login Report on login activity by user, group, or terminal

To access these functions, make the following selection from within **sysadmsh**:

Accounts→Report

The Reports selection can generate a variety of reports. You can use the reports for security purposes (for example, listing parameters in the Protected Password and Terminal Control databases). Because these reports show system and peripheral usage, you may find them useful to fine-tune and reconfigure the system.

For all the reports, upon executing the screen you are asked to direct the output to the screen, the printer or a file.

You can filter screen output through any of the system pagination programs. The **more**(C) program is set up as the default. For printer output, you can name the printer device; if you don't name it, the system default printer destination is used. If redirecting output to a file, use full pathnames. No matter what category of report you select, you are always requested to define how you want the data displayed: on screen, on printer, or into a file.

Reporting Password Status

The first report selection, *Password Database*, reports on the account and password parameters set up for accounts. The report is derived from information in the Protected Password database. Password status can be reported in several categories:

Impending	Reports on accounts with passwords about to expire
Expired	Reports on accounts with expired passwords
Dead	Reports on accounts with dead passwords
User	Report on a single user
Group	Report on a single group of users
Full	List all entries in password database.

The *Impending* option reports on accounts that have, or will soon have, expired passwords. This includes all accounts with already-expired passwords as well as those that will expire within one week. Although an impending expiration is not an error per se, this report lets you see users that wait until the last moment to change passwords. You may want to revise the system-wide and per-user password expiration periods based on information obtained here.

The *Expired* option reports on all accounts with expired passwords. These may or may not be dead passwords. All such accounts need some administrative action before the account is usable; minimally, the password must be changed.

The *Dead* option reports on those accounts whose password lifetime has expired, which causes the account to reject further logins.

The *User* option reports on the individual user that you specify. Enter the user's login name to activate it.

The *Group* option reports on a single specified group. This report includes all the users that belong to the specified group (as shown in the *Group Membership* field of the *User Account Maintenance Screen*).

Finally, the *Full* option reports statistics for all users on the system.

The reports use the following abbreviations:

Dflt Default.

Y, N, D Yes, No, Default. Some selections have three possible values: yes, no, and the default value used by the system, which can be either.

Example Report: Group

The following is a sample report on the password activity of group "hamster." The abbreviations under "Password Parameters" correspond to the system-wide default password parameters.

```
                    Password Database Report
                         System  unix
                    Wed Mar 22 10:56:29 1989

                       Password Parameters
[1] User Name Type      Min  Exp  Life Rnd? Pck? Rst? Lck?
    ---- ---- ----      ---  ---  ---- ---- ---- ---- ----
          Last Changes                Last Logins        Consec
[2] Success     Failed        Success      Failed       #Failed
    -------     ------        -------      ------       -------

[3] Kernel Authorizations
    ------ --------------
[1] alvin     general  Dflt Dflt Dflt D    D    D    Y
[2] 03/22/89    NEVER       03/22/89    NEVER       -
[3] DEFAULT
[1] simon     general  Dflt Dflt Dflt D    D    D    N
[2] 03/22/89    NEVER       03/22/89    NEVER       -
[3] DEFAULT
[1] theodore  general  Dflt Dflt Dflt D    D    D    N
[2] 03/22/89    NEVER       03/22/89    03/22/89    -
[3] DEFAULT
```

Chapter 10

UNIX Directories and Special Device Files

This chapter lists the most frequently used files and directories on a UNIX system. Many of these files and directories are required for proper operation and must not be removed or modified. The following sections briefly describe each directory.

This chapter also contains information needed to create device nodes relating to filesystems and terminals. For a full description of the special files mentioned here, see the manual pages in section (HW).

UNIX Directories

The following subsections discuss each of the main directories of the operating system.

The Root Directory

The root directory (/) contains the following system directories:

/bin	UNIX command directory
/dev	Device special directory
/etc	Additional program and data file directory
/lib	C program library directory
/mnt	Mount directory (reserved for mounted filesystems)
/usr	User service routines (may contain user home directories)
/tcb	System files that are part of the TCB (Trusted Computing Base)
/tmp	Temporary directory (reserved for temporary files created by programs)

All of the above directories are required for system operation.

The root directory also contains a few ordinary files. Of these files, the most notable is the *unix* file which contains the UNIX kernel image.

The */bin* Directory

The */bin* directory contains the most common UNIX commands, that is, the commands likely to be used by anyone on the system. Here is a sample list of programs in */bin*:

basename	echo	passwd	su
cp	expr	rm	sync
date	fsck	sh	tar
dump	login	sleep	restor
dumpdir	mv	stty	test

These commands and all others in the */bin* directory are required.

The */dev* Directory

The */dev* directory contains special device files which control access to peripheral devices. All files in this directory are required, and must not be removed. There are several subdirectories to the */dev* directory. Each of these subdirectories holds special device files related to a certain type of device. For example, the */dev/dsk* directory contains device files for floppy and hard disks. The operating system supports both XENIX and UNIX device naming conventions. Where appropriate, the files in the */dev/dsk* directories are linked to the device files that exist in */dev*. You can access the same device through the file in */dev* or the file for the same device in a subdirectory of */dev*.

Table 10.1 contains a partial list of devices.

Table 10.1.

/dev Device Nodes

UNIX Device	XENIX Device	Name
/dev/console	same	System console
/dev/rdsk/*	/dev/r*	Raw devices
/dev/dsk/0s0	/dev/hd00	Entire disk on drive 0
/dev/dsk/0s1	/dev/hd01	First disk partition on drive 0
/dev/dsk/0s2	/dev/hd02	Second disk partition on drive 0
/dev/dsk/1s0	/dev/hd10	Entire disk on drive 1
/dev/dsk/1s1	/dev/hd11	First disk partition on drive 1
/dev/dsk/1s2	/dev/hd12	Second disk partition on drive 1
/dev/dsk/f05d9	/dev/fd048ds9	360K floppy drive 0
/dev/dsk/f05q	/dev/fd096ds9	720K floppy drive 0
/dev/dsk/f05h	/dev/fd096ds15	1.2MB floppy drive 0
/dev/dsk/f03h	/dev/fd0135ds18	1.44MB floppy drive 0
/dev/lp	same	Lineprinter
/dev/kmem	same	Kernel virtual memory
/dev/mem	same	Physical memory
/dev/null	same	Null device
/dev/rmt0	/dev/rct0	QIC tape device
-	/dev/rft0	QIC-40 tape device
-	/dev/rctmini	Mini-cartridge tape device
/dev/root	same	Root file structure
/dev/swap	same	Swap area
/dev/tty*nn*	same	Terminals

The */etc* Directory

The */etc* directory contains miscellaneous system program and data files. All files are required, but many can be modified.

/etc/mnttab	Mounted device table
/etc/mount	For mounting a file structure
/etc/mkfs	For creating a file structure
/etc/init	First process after boot

The following data files may be modified, if desired. No files may be removed.

/etc/rc	Bootup shell script
/etc/rc0	Shutdown shell script
/etc/rc2	Bootup shell script
/etc/termcap	Terminal capability map
/etc/motd	Message of the day

The data files in the directories */etc/rc.d* and */etc/rc2.d* contain initialization commands run by the **/etc/rc2** script when the system goes into multiuser mode.

The data files in the directory */etc/default* contain default information which is used by system commands (see **default**(F)). The following data files may be modified. No files may be removed.

Table 10.2.

/etc/default Files

File	Utility
/etc/default/archive	**sysadmsh**(ADM) backup default information
/etc/default/authsh	**sysadmsh**(ADM) default account information
/etc/default/backup	**backup**(ADM) default information
/etc/default/boot	**boot**(ADM) information
/etc/default/cleantmp	**cleantmp**(ADM) default information
/etc/default/cron	**cron**(C) default logging information
/etc/default/dumpdir	**xdumpdir**(ADM) default information
/etc/default/filesys	**sysadmsh**(ADM) default filesystem information
/etc/default/format	**format**(C) default information
/etc/default/goodpw	**goodpw**(ADM) default password check information
/etc/default/idleout	**idleout**(M) default information
/etc/default/lang	default locale information
/etc/default/lock	**lock**(C) default information
/etc/default/login	**login**(M) default information
/etc/default/lpd	**lp**(C) default information
/etc/default/man	**man**(C) online man page default information
/etc/default/mapchan	**mapchan**(M) default information
/etc/default/micnet	**micnet**(M) default information
/etc/default/msdos	Location of DOS disks (A:, B:,...)
/etc/default/passwd	**passwd**(C) default information
/etc/default/purge	**purge**(C) default information
/etc/default/restor	**xrestore**(ADM) default information
/etc/default/su	**su**(C) default information (note that you must create this file yourself.)
/etc/default/tape	**tape**(C) default device information
/etc/default/tar	**tar**(C) default device information
/etc/default/usemouse	**usemouse**(C) default information

The */lib* Directory

The */lib* directory contains runtime library files for C and other language programs. The directory is required.

The */mnt* Directory

The */mnt* directory is an empty directory reserved for mounting removable filesystems.

The */tmp* Directory

The */tmp* directory contains temporary files created by UNIX programs. The files are normally present when the corresponding program is running, but may also be left in the directory if the program is prematurely stopped. You may remove any temporary file that does not belong to a running program.

The */usr* Directory

The */usr* directory consists of several subdirectories that contain additional UNIX commands and data files. It is also the default location of user home directories.

The */usr/bin* directory contains more UNIX commands. These commands are less frequently used or considered nonessential to UNIX system operation.

The */usr/include* directory contains header files for compiling C programs.

The */usr/lib* directory contains more libraries and data files used by various UNIX commands.

The */usr/spool* directory contains various directories for storing files to be printed, mailed, or passed through networks.

The */usr/tmp* directory contains more temporary files.

The */usr/adm* directory contains data files associated with system administration and accounting. In particular, the */usr/adm/messages* file contains a record of all error messages sent to the system console. This file is especially useful for locating hardware problems. For example, an unusual number of disk errors on a drive indicates a defective or misaligned drive. Since messages in the file can accumulate rapidly, the file must be deleted periodically.

The */tcb* Directory

The */tcb* directory contains all files that are part of the TCB (Trusted Computing Base). These files comprise the security enhancements made to the operating system to make it more secure than other UNIX operating systems.

Log Files

A variety of directories contain log files that grow in size during the normal course of system operation. Many of these files must be periodically cleared to prevent them from taking up valuable disk space. Table 10.3 lists the files (by full pathname) and their contents.

Table 10.3.

System Log Files

Filename	Description
/etc/ddate	Records date of each backup.
/usr/adm/pacct	Records accounting information; grows rapidly when process accounting is on. (See **accton**(ADM) and **acccom**(ADM).)
/usr/adm/messages	Records error messages generated by the system when started. (See **messages**(M).)
/etc/wtmp	Records user logins and logouts. (See **login**(M).)

(*Continued on next page.*)

Table 10.3.

System Log Files (*Continued*)

Filename	Description
/usr/adm/sulog	Records each use of the **su** command; grows only if option is set in the */etc/default/su* file. You must create */etc/default/su.* (See **su**(C).)
/usr/lib/cron/cronlog	Records each use of the **at**(C) and **cron**(C) commands.
/usr/spool/micnet/remote//LOG*	Records transmissions between machines in a Micnet network. The (*) must be the name of a remote machine connected to the current machine.
*/usr/spool/uucp/.Log/utility/sitename/**	Logs UUCP commands used over a UUCP network. The *utility* and *sitename* are the is the name of the UUCP utility, and the name of the remote site, respectively.
*/usr/spool/uucp/.Log/.Old/**	Old log files are stored in this directory by the **uudemon.clean** shell script.

Special Device Files

Many of the filesystem maintenance tasks described in this guide require the use of special filenames, block sizes, and gap and block numbers. The following sections describe each in detail.

Special Filenames

A special filename is the name of the device special block or character I/O file, which corresponds to a peripheral device, such as a hard or floppy disk drive. These names are required in such commands as **mkfs**(ADM), **mount**(ADM), and **df**(C) to specify the device containing the filesystem to be created, mounted, or searched.

Table 10.5 lists the XENIX and UNIX special filenames and corresponding devices, for hard and floppy disk drives on a typical computer.

Table 10.4.

Device Special Filenames - Disks

Filename	Disk Drive
/dev/fd0	Floppy Drive 0
/dev/dsk/f0	Floppy Drive 0
/dev/fd1	Floppy Drive 1
/dev/dsk/f1	Floppy Drive 1
/dev/hd00	Entire hard disk
/dev/dsk/0s0	Entire hard disk
/dev/root	Root filesystem
/dev/u	User filesystem

Block Sizes

The block size of a disk is the number of blocks of storage space available on the disk, where a block is 1024 bytes of storage. Most commands report disk space in terms of 512 byte blocks, in particular **df**(C), **du**(C), **lc**(C), and **find**(C). A 500 byte file on a 1024 byte block filesystem is reported as using 2 blocks by these utilities, as the file uses one system block which is equivalent to two 512 byte blocks. The size of a 40 megabyte hard disk in 1024 byte blocks is 39168. Note that some of the blocks on the disk are reserved for system use and cannot be accessed by user programs. The block size of a typical floppy disk depends on the total storage capacity of the disk, as given by the manufacturer.

Gap and Block Numbers

The gap and block numbers are used by the **mkfs**(ADM), and **fsck**(ADM), commands to describe how the blocks are to be arranged on a disk. Table 10.6 lists the gap and block numbers for the floppy and hard disks used with a typical computer.

Table 10.5.

Gap and Block Numbers

Disks	Gap	Block
Floppy Disk, 48ds9	1	9
Floppy Disk, 96ds15	1	15
Floppy Disk, 135ds9	1	9
Floppy Disk, 135ds18	1	18
Hard Disk	1	34

The number of blocks can also be determined by multiplying the number of sectors per track (usually 17) by the number of heads on the hard disk, dividing by 2 (since there are 2 blocks per sector), and rounding off to the nearest integer.

Terminal and Network Requirements

The **enable** and **disable** commands are used to add and remove terminals on a system. The install option of the **netutil** program is used to build a network. The preceding commands and option require the names of the serial lines through which a terminal or network is to be connected. The following table lists the device special filenames of the two serial lines (actually two serial ports either with or without modem control). The character I/O files corresponding to these serial lines can be found in the */dev* directory. Note that the files */dev/console* and */dev/tty01* through */dev/tty12* represent "hardwired" devices and are not available for connection to terminals or hardware. Also, refer to **serial**(HW) for more information on serial lines.

Table 10.6.

Serial Devices

Filename	Line
/dev/tty1a	main serial line (without modem control)
/dev/tty2a	alternate serial line (without modem control)
/dev/tty1A	main serial line (with modem control)
/dev/tty2A	alternate serial line (with modem control)

Chapter 11

Adding Ports and Modems

One important task of the system administrator is to add peripheral devices such as modems to the system. Adding these serial devices lets more users access the system and adds to overall system capabilities.

This chapter explains the following tasks:

- Physically connecting serial devices to your computer
- Maintaining serial devices

Note that physical connections between a device and the system vary according to hardware configuration. For specific information about connecting your serial device, refer to the hardware manuals provided with the device and with your computer.

Adding and Configuring Serial Ports

To add a multi-port expansion card, you must first determine whether the card is a "smart" serial card or a standard serial card. If the card is a "smart" card, the manufacturer will have supplied installation software and a driver. These should be all you need to add the card to your UNIX system.

Before installing your card, check your *Release Notes* for information about hardware compatibility. Follow the instructions for insertion furnished with your card, referring to your computer hardware manual if necessary.

If your card is a standard serial card, the following instructions explain how to create new device files for additional ports.

1. First, return to the desktop and double-click on the Sysadmsh icon to bring up the main **sysadmsh**(ADM) menu. Then make the following selection:

   System→Hardware→Card_Serial

2. The following is displayed:

```
You would like to install a:
        1. 1 port card
        2. 2 port card
        3. 4 port card
        4. 5 port card
        5. 8 port card
Select an option or enter 'q' to quit:
```

   Enter the appropriate number and press **Return.**

3. The program responds with the following menu (only COM1 and COM2 appear and are usable on most systems):

```
The card is configured as:
        1. COM1
        2. COM2
        3. COM3
        4. COM4
Select an option or enter 'h' for help or 'q' to quit:
```

   If you select "h", you see a table listing ports, card types, I/O addresses, and status addresses.

   Enter a number and press **Return.** After **mkdev** accepts the COM slot, you see a list giving the newly configured ports and their modem control counterparts. For example, tty2a and tty2A refer to the same serial port, but tty2A has modem control, whereas tty2a refers to the same port without modem control. You can access the port by only one name at a time, either with or without modem control.

Now that your serial ports are configured, make sure that they are also defined in the system hardware configuration.

Check your computer hardware manual to determine how your system is configured. If your system is configured using a CMOS database, the ports are defined in the database (see **cmos**(HW)).

If your system is configured with switch settings on the main system board, define the new ports by setting the proper switches (refer to your hardware manuals for the settings).

NOTE: An error message is displayed if you attempt to access a serial port that has not been installed and defined.

Using a Modem on Your System

This section explains how to connect and use a modem on your UNIX system.

Serial Lines

The operating system supports modem control on serial ports. Table 11.1 contains sample device names of serial ports with and without modem control.

Table 11.1.

Serial Lines

Device	Function
/dev/tty1a	main serial adapter **without** modem control.
/dev/tty1A	main serial adapter **with** modem control.
/dev/tty2a	alternate serial adapter **without** modem control.
/dev/tty2A	alternate serial adapter **with** modem control.

/dev/tty1a and */dev/tty1A* refer to the same serial port (likewise for */dev/tty2a* and */dev/tty2A*). The operating system uses different device-driver subroutines for each. Never attempt to use both modem and non-modem control ports at the same time or you will see the warning:

```
cannot open: device busy
```

For systems including multi-port serial cards, the devices */dev/tty[1,2][a-m]* refer to use *without* modem control, and the devices */dev/tty[1,2][A-M]* refer to use *with* modem control.

Dialing Out From Your Computer

The **cu**(C) and **uucp**(C) utilities are used to call remote systems and transfer data on UNIX systems. The file */usr/lib/uucp/Devices* (referred to as **Devices**) contains information used by these programs to determine the characteristics of a particular serial line.

The **Devices** file contains lines which specify the device for the line, the call-unit associated with the line, and the baud rate, that are to be used by UUCP. (Modem control devices should be used with lines connected to modems.)

Using Dialer Programs

For dialing, both **cu** and UUCP use a common set of dialers, which can be standalone binaries (programs) like */usr/lib/uucp/dialHA12*, or entries from the file */usr/lib/uucp/Dialers*. (For more information on *Dialers* file entries, see *Using ODT-NET*.)

The source for several dialer programs and a makefile for recompiling the source program are included in the directory */usr/lib/uucp*. If you have any other kind of modem, you can modify any of the source files and create your own dialer program. Note that you must have the *Development System* installed to compile a program.

To make a new dial program, follow these steps:

1. Change directory to */usr/lib/uucp* with the following command:

   cd /usr/lib/uucp

2. Edit the file *makefile* in the directory */usr/lib/uucp* and find the line that reads:

   ```
   EXES= dialHA12 dialHA24 dialTBIT dialVA3450
   ```

   and add the name of the dialer program you wish to use. When this is done, exit the file, saving the changes you have made.

3. Next, enter the command:

   make

   to your shell prompt and press **Return**.

4. When the **make** command is finished, you have a new dialer program. This can be used in the fifth field of an entry in the **Devices** field.

Installing a Dial-out Modem

NOTE: Internal modems are not recommended. This is because problems with such modems are difficult to debug. There are sometimes interrupt conflicts that cannot easily be resolved.

When you are hooking up your modem, or any other device, make sure that serial wires connected to your computer are not left hanging. An unterminated line connected to your computer can considerably reduce system performance; always unplug a modem wire at the computer end instead of at the modem end.

Three-wire cables often used to connect terminals to the computer are not sufficient for connecting modems. For a modem cable on a 25-pin serial port, pins 2, 3, 7, 8, and 20 must be connected straight through. If you are unsure as to what to use, a cable that connects all pins will work correctly. A ribbon cable will do, or what is called a "straight-through" cable, meaning that it connects the pins straight across.

To install your modem, follow these steps:

1. Make sure the UUCP package is installed. Use **custom**(ADM) to install if necessary.

2. Make sure the serial port you have chosen for your dial-out modem is recognized at boot up and, if the modem is internal, make sure the COM port the internal modem is configured for does not conflict with any other device. Only serial devices attached to COM1 and COM2 are supported.

3. Make sure the port is disabled by entering:

   disable *ttyname*

4. Connect the modem to the machine using a "straight-through" cable (pins 2 & 3 not crossed). The cable must have at least pins 2, 3, 7, 8, and 20 connected.

   Most standard COM ports use "straight-through" cables (meaning all pins are connected straight across the cable), but some hardware requires a null-modem cable (pins 2 & 3 crossed). A standard COM port is known as DTE, a port that needs a null-modem cable is known as DCE. Check your hardware documentation if you are not sure. If the COM board is a DCE, you will need a null-modem cable.

5. Add the correct entries to the */usr/lib/uucp/Devices* file. This file should have two entries for each serial port being used for a modem. One of the entries is used when you start a call using the modem (the ACU line), and the other line is used to configure the modem using the standard Hayes command set (the Direct line). You should use an entries like these, which are set up for a Hayes-compatible modem operating at 2400 baud, using COM1:

```
Direct  ttyla - 1200-2400 direct
ACU  ttylA - 1200-2400 /usr/lib/uucp/dialHA24
```

   Make sure that the entries do not have a pound sign ("#") in front of them. This is the syntax to show that the line is only a comment, and is to be ignored. There are many examples in the **Devices** file that are commented out with this character.

6. Enter the following command to establish UUCP as the owner of the port you have selected:

   chown uucp /dev/*ttyname*

7. Test the dial-out modem. To test the modem's ability to dial correctly use the following command:

   cu -ltty1a dir

   You should see a message indicating that you are connected. If you see the message "cu: dir permission denied," the user executing the **cu** command does not have write permission on the */usr/lib/uucp/Devices* file. If you do not see such a message, and there was no message to indicate that you connected correctly, either the **cu** command is incorrect, the **Devices** file is incorrect or the serial port is not operating correctly.

   NOTE: The instructions that follow assume a Hayes-compatible command set and response codes. Other modems may use other conventions. Consult your modem documentation for further details.

   If you do see a message confirming your connection, enter

   AT

   on your keyboard. "OK" should be echoed back onto your computer screen. If the modem is set to return result codes as numeric codes rather than text, you will see a 0. If this does not occur, check that the "receive" light on the modem

flashes when you press a key. This indicates the modem is receiving signals from the keyboard. If this light is not flashing, check your cable and modem switch settings. If the "receive" light flashes, but you still don't get an "OK" response from the modem, try entering

ATE1

at your computer keyboard to enable the modem's echo capability.

If you get the expected responses, you can dial out by entering

ATDT *phonenumber*

Once you have confirmed that the modem can dial out, exit **cu** by entering:

~.

and then press **Return**.

8. You are now ready to dial into another system. You should use the following command to dial out:

   cu -ltty1A 555-1212

   You should change "555-1212" to the phone number of the system that you want to dial. If you have any problems, refer to the next section on troubleshooting your dial out modem. If the line is also to be used for dial-in, follow the additional steps specified in "Installing a Dial-in Modem" in this chapter.

Troubleshooting Your Dial-out Modem

The examples below assume a modem attached directly to COM1. Other serial ports are often used. If you have problems, first verify that the phone jack is plugged in and you have a dial tone on the phone line.

1. **Problem:** In testing the modem connection with the command:

   cu -s1200 -ltty1a dir

   I get a connected message but when I type "AT" there is no "OK" message.

Remedy A: Check the cable and modem switch/software settings. If using a straight through cable try a null modem cable using at least pins 2, 3, 7, 8 and 20. After issuing the **cu** command, watch the lights on the modem and hit the <Return> key several times. The "receive" light should flash as you hit the key. If it doesn't, you need to check your cable to make sure that pin 2 is connected correctly (pin 2 is the data transmission line from the serial port to the modem). If the "receive" light flashes, try using ATE1 to turn on the modem's echo capability.

Remedy B: The serial port on the computer may be defective. Try attaching the modem to a different serial port, or attach a terminal or serial printer to the port to confirm that it is functioning. If the port is not functioning, check your hardware documentation for an appropriate repair facility.

Remedy C: The modem may be defective. If this is the case, check your hardware documentation for an appropriate repair facility.

2. **Problem:** Modem dials but call never connects.

   Remedy A: The phone number may be incorrect or not operational, or the phone line to which the modem is attached may be faulty.

   Unplug the modem from the telephone line, and plug in a regular telephone. Try dialing the number yourself to make sure the modem on the other end of the line is answering the call.

   Remedy B: Listen carefully to your modem while it dials the call. Some business phone systems require that there be a pause between certain numbers. number.

   A hyphen is used in the **cu** command to indicate a pause of two seconds, for example: "9----458--1234".

   The hyphen given to the **cu** command is translated by the dialer into the appropriate code for your modem. For a Hayes-compatible modem, it's translated to a comma before being sent to the modem.

3. **Problem:** In dialing out you see the message:

   ```
   Connect failed: NO DEVICES AVAILABLE
   ```

   Remedy A: There is no entry in the **Devices** file for the modem port. Here are example entries for a Hayes-compatible modem running at 2400 baud on */dev/tty1A*:

   ```
   Direct  tty1A - 2400 direct
   ACU  tty1A - 300-2400 /usr/lib/uucp/dialHA24
   ```

Make sure that there is no pound sign ("#") at the beginning of these lines in the **Devices** file.

Remedy B: The modem port in **Devices** does not have the correct baud rate associated with it. Make sure that if you use the option to **cu** to specify the baud rate that there is an entry in **Devices** that corresponds to that baud rate.

4. **Problem:** Modem answers but I get garbage characters on my terminal.

   Remedy A: The site you are calling may be set to different data bit and parity values than you are using. By default **cu** uses 8 data bits, and no parity. Use **cu -e** for 7 data bits, even parity, and **cu -o** for 7 data bits, odd parity.

   Remedy B: The remote computer is at a different baud rate. If you are dialing in to another UNIX system, send a break signal to cause the remote site to switch baud rates during the login sequence. Always have the system start at the highest baud rate and move down as necessary. To send the break signal, enter:

   ~%b

   Remedy C: There may be noise on the line. This becomes more acute when operating at 2400 baud or higher. Check your phone line. Normally, when there is a problem with line noise you will see garbage characters appear on the screen continually, as if there was a system on the other end of the line trying to send valid data.

5. **Problem:** My modem does not hang up at the end of a call.

   Remedy A: A non-modem control port is being used. Non-modem control ports should only be used with terminals, and when configuring the modem. Which port to use is configured in the **Devices** file. Change the non-modem control serial port you specify to the corresponding modem control port. For example, the modem control port associated with tty1a is tty1A.

   Remedy B: The CD (Carrier Detect) light on the modem doesn't go off when the call is disconnected. Check the modem switches to verify that the modem is set to detect the incoming carrier or, if this is a Hayes 2400 or compatible modem, use the AT&C1 command.

   Remedy C: The modem is not set to detect DTR (Data Terminal Ready). Check the modem switches to verify that the modem is set to detect DTR or, if this is a Hayes 2400 modem, use the AT&D2 command. Some modems have a switch that can be set to ignore DTR, and this switch should not be on.

Dialing Into Your Computer

To allow dialing into your computer, you must enable a serial line that recognizes modem control signals, with the **enable**(C) command.

To use the main serial adapter (COM1), enter:

```
disable tty1a
enable tty1A
```

Or, for the alternate serial adapter (COM2), enter:

```
disable tty2a
enable tty2A
```

Note that tty1A and tty1a refer to the same (main) serial line, and tty2A and tty2a refer to the same (alternate) serial line. Do not use the same line in both its modem and non-modem modes at the same time as this will cause an error.

Installing a Dial-in Modem

The following procedure provides step by step instructions for installing a modem for dial-in operation. (Passwords are recommended for dial-in lines; refer to the section on "Adding Dial-in Password Protection" in the "Maintaining System Security" chapter for details.)

1. Follow the steps for installing a modem for dial-out. This ensures that you have a working hardware connection.

2. Some modems have switches or software commands for setting the modem configuration. If your modem has such settings, configure it as follows, following the instructions in your modem manual.

   NOTE: If the modem is to be shared between dial-in and dial-out, Step 3 can be omitted. Initialization to dial-in is automatically performed whenever the system comes up, or a dial-out completes.

3. Set the modem to automatically answer the phone when a call comes in.

   Most internal modems do not have auto-answer and some external modems do not have this setting. If this is the case, place the following line in the initialization file */etc/rc.d/8/userdef*:

   (stty 1200; echo "ATS0=1\r" >/dev/tty1a) </dev/tty1a

   "tty1a" should be changed to match the non-modem control device the modem is connected to. "1200" should be changed to the highest baud rate used by the modem. "ATS0=1" is the command to put Hayes compatible modems into autoanswer mode. The "\r" is needed to send a carriage return signal to the modem to terminate the command line.

4. Set up your modem so that it does not answer when the DTR line is not active and it disconnects from the current connection when DTR goes from active to inactive.

5. The CD line should be set to follow the incoming carrier, i.e. low when a carrier is present, high when a carrier is not present.

6. Set up your modem so that it does not echo commands or display responses.

7. Make sure the port is disabled by entering the command:

   disable *ttyname*

   Where *ttyname* is the non-modem control port.

8. Select the desired **gettydefs** entry in the */etc/inittab* file. Entry "2" will select the 1200-2400-300 cycle.

9. Enable the port you are using for your modem with the following command:

   enable *ttyname*

   Where *ttyname* is the modem control port.

10. Dial this modem from another modem.

11. If you are unable to successfully dial-in, see the next section on trouble shooting your dial-in modem.

Troubleshooting Your Dial-in Modem

The examples below assume that your modem is attached directly to the COM1 port. In practice, a modem may be attached to other serial ports.

1. **Problem:** Modem is not answering phone.

   Remedy A: The modem serial port is not enabled.

   Enter the following commands:

   disable /dev/tty1a
   enable /dev/tty1A

   Remedy B: The modem is not configured to auto-answer.

   Check your modem switches or, if this is a Hayes 2400 modem, use the appropriate modem software command (See "Hayes Modem Settings" at the end of this section for Hayes commands). Enter **cu -ltty1a dir** to the modem and use the command "ATS0=1" to set auto-answer.

   Remedy C: The DTR (Data Terminal Ready) line is not connected from the computer to the modem.

   Check Pin 20 and make sure it is connected. Pins 2, 3, 7, 8, and 20 are used for modem hookup.

2. **Problem**: Modem answers, but hangs up immediately upon connection.

   Remedy: The modem is set to autoanswer and to detect DTR, but the DTR line is not asserted. Check the following possibilities:

   a) The modem control port may not be enabled. Enter the command:

   disable /dev/tty1a
   enable /dev/tty1A

   b) The cable is incorrect.

   If you are using a "straight through" cable with at least pins 2, 3, 7, 8 and 20 connected, check that pin 20 (DTR) is properly connected.

3. **Problem**: I see the error message "Garbage or loose cable on /dev/tty1A, port shut down" on my console when a call comes in to my modem.

   Remedy A: Your modem is set to echo back data or send responses to commands.

   It is very likely that the modem is sending a "RING" signal to indicate that the phone you are calling is ringing. Since the CD signal is not active, getty interprets this as random data on the serial line. To correct this, set the modem to turn off echo and not to send command responses. The proper Hayes 2400 modem command is "ATE0Q1".

   Remedy B: If you have an internal modem and the above options don't eliminate the error message, it is likely that you have an incompatible modem. Try replacing your modem with a standard Hayes-compatible model.

4. **Problem**: The modem answers, but I get no login prompt.

   Remedy A: The CD line is not being asserted by the modem after the modem has answered the phone.

   Check the switches on your modem or, if this is a Hayes 2400 modem, use the appropriate modem software command.

   Remedy B: The port is not enabled. Enter the command:

   enable /dev/tty1A

   Remedy C: An incorrect */etc/gettydefs* entry is being used and is selecting the wrong baud rate.

   Check the modem port device in the */etc/inittab* file. It will appear similar to the following example:

   ```
   t1A:2:respawn:/etc/getty tty1A m
   ```

   The last character on the line is the pointer to the entry in the */etc/gettydefs* file. Check this entry to make sure that it has the correct settings.

5. **Problem**: The screen scrolls uncontrollably when I log in, usually displaying a series of login prompts.

   Remedy: The modem and non-modem devices are both enabled.

   Disable the non-modem device by entering the command:

   disable /dev/tty1a

6. **Problem**: I get a login prompt but nothing coherent afterward.

   Remedy: The line settings are incorrect.

   Find out what the serial line settings are on the system that you're calling into. The standard settings that **cu** uses are eight data bits, one stop bit, and no parity. If the remote system is using even parity, use the **-e** option to **cu**, which selects even parity, or the **-o** option which select odd parity.

   If you're dialing into a UNIX system, check the */etc/inittab* file on the remote system to verify that the "pointer" into the **gettydefs** file is correct. Chances are that the serial line characteristics don't match between the stty settings defined in the third field of the selected **gettydefs** entry. Try changing the port's setup to 8 data bits, one stop bit, and no parity. The entry should look something like this:

   4 # B1200 HUPCL # B1200 CS8 SANE HUPCL TAB3 ECHOE IXANY #\r\n@!login: # 5

Shared Dial-in/Dial-out

The Operating System supports the use of dial-in and dial-out on the same modem line, without having to disable the login.

When a dial-out program is using the line, the login will be disabled. If someone is logged in on a line when a dial-out program attempts to use it, the dial-out program will fail to lock the device.

For this feature to work correctly, the modem control device must be used, and the modem must set CD to high when a carrier is present and low when a carrier is not present. (For information on using dial-in/dial-out in conjunction with **uucp**, refer to the "Building a Remote Network with UUCP" chapter in this guide.)

Installing a Shared Dial-in/Dial-out Modem

The following procedure allows you to install a shared dial-in/dial-out modem.

1. Perform the steps of installing a modem for dial-out, and those for installing a dial-in.

2. To dial out, simply invoke **cu** with the appropriate options. The getty on the line will automatically be suspended for the call out, and restarted when the call is completed.

Hayes Modem Settings

Proper modem configuration is necessary when using **cu** and **uucp**. Modem settings differ for each modem. Consult your modem manual for the proper switch settings.

Smartmodem 1200

If you have a Hayes Smartmodem 1200 or compatible, switches 3 and 8 should be down:

	1	2	3	4	5	6	7	8
up	●	●		●	●	●	●	
down			●					●

When switch 3 is down, the resulting codes will be sent to, (echoed by), the modem to the terminal or computer. When switch 8 is down, the modem is able to interpret the command being issued. This allows both the UNIX and DOS communications systems to work.

Smartmodem 2400

The Hayes 2400 Smartmodem or compatible modem requires on-line configuration if it is to be used as a dial-in line. Note that the Hayes 2400 does not answer the phone with a 2400 baud carrier if it is not set up with 2400 baud commands. You must configure the modem by issuing set up commands via **cu**(C). The form of the **cu** command is:

cu -s2400 -ltty*nn* **dir**

nn is the "tty" number of the serial line. To configure a modem on tty1A, enter this command and press **Return**.

cu -s2400 -ltty1A dir

Next, enter the following commands to configure the modem. They will be saved in the modem's non-volatile memory. If you do not want to save the settings, do not enter the last command (at&w). Commands are in the left column and short descriptions of what they do are in the right column. Follow each command with a **Return**:

AT&f	Fetch factory configuration.
ATT	Tone dialing.
ATl0	Low speaker volume.

AT&d2	Set dtr "2": go on hook when dtr drops.
AT&c1	Set dcd "1": dcd tracks remote carrier.
ATs0=1	Answer phone after 1 ring (AA light should come on).
ATs2=128	Disable modem escape sequence.
ATe0	No echo (modem will no longer echo what is sent to it).
ATq1	Quiet mode (modem will not respond with "OK" after this command or any that follow).
AT&w	Saves settings in non-volatile memory.

Exit from **cu** by entering a "tilde" and a "period", followed by a **Return**:

```
~.
```

The modem is now configured and ready for use.

Chapter 12
Using Printers

Printers are a highly important attachment to any computer system. Most systems require the ability to print out data on paper. A wide variety of printing hardware or line printers are supported. Some line printers are *parallel* devices, but most are connected as *serial* devices.

To add a printer, the system administrator must:

- connect the physical hardware to the computer, then
- use the correct system commands to enable the printer for operation.

This chapter explains how to do this and how to maintain printers once they are added. Note that physical connections between a printer and the system vary depending on hardware configuration. This chapter provides some information about making the necessary physical connections, but for more information about these connections, see the hardware manuals provided with the printer and your computer.

The operating system supports serial printers that use the standard RS-232 interface. To find out if your printer uses this interface, check your hardware documentation. RTS/CTS protocols are also supported.

The Printer Spooling System

The UNIX line printer spooling system is a collection of commands that help you, as system administrator, to install, monitor, and control efficiently the line printers serving your system. A request to print a file is *spooled* or lined up with other printing jobs to be sent to the printer. Each print job is processed and waits its turn in line to be printed, thus the term *queue*.

When a user requests a file to be printed using the **lp**(C) command, the line printer system responds with a "request ID." This consists of the name of the printer on which the file is printed and a unique number identifying the file. With this request ID, the user can find out the status of the print request or cancel it. The **lp** options help the user to control printer output easily.

The print service performs the following functions:

- handles the task of receiving files users want printed,
- filters the files (if needed) so they can print properly,
- schedules the work of one or more printers,
- starts programs that interface with the printer(s),
- keeps track of the status of jobs,
- alerts you to printer problems,
- keeps track of the mounting of forms, and
- Issues error messages when problems arise.

There are several terms used in this chapter to describe the operation of the print service:

device The target for **lp** output. It can be a hard-wired printer, a terminal that is sometimes used as a printer, or a regular file. A device can be represented by a full pathname.

printer The name assigned by the system administrator to represent a device. This name can have up to 14 characters. At different times, a printer can be associated with different devices.

class An ordered list of printers. Print requests sent to a class of printers are printed by the first available member of that class.

destination A place where print requests are sent. A destination can be a class or a printer.

Consult your computer and line printer hardware manuals for information on making the connection between your system and printing devices.

Installing a Printer

This section instructs you on how to install new printing devices on your UNIX system. You must connect the printer to a proper port (serial port for serial printers, parallel port for parallel printers), ensure that it works, and set up the UNIX printer spooling software using the **sysadmsh** "Printers" selection. Follow the steps below to install a printer:

1. Find a place for your printer and make sure that it is properly assembled and plugged in to a power outlet.

2. **If you are connecting a serial printer:** connect the RS-232 cable from your computer serial port to the port on your printer. Serial printers must be capable of supporting XON/XOFF or DTR protocols and must be configured for those protocols. Next, enter the following command substituting the correct port number for *nn*:

   disable /dev/tty*nn*

   Press **Return**. This disables logins on the port you have connected to your printer and allows the port to be used for serial communication.

3. **If you are connecting a parallel printer:** The printer must use a standard Centronics interface cable. The parallel port on a monochrome card should be configured for interrupt vector 7, and it is recognized as **lp1** when booting up. The main parallel port should be configured for interrupt vector 7 and is recognized as **lp0**. You must use either the main or the monochrome port - not both - to avoid a hardware conflict. The alternate or second parallel port should be configured for interrupt vector 5, and it is recognized as **lp2**. Make sure no other hardware is using these interrupts. (See your hardware manual for information on configuring your parallel ports.)

4. Verify that you have correctly hooked up the printer by sending data directly to the device. Enter one of the following commands:

   For serial printers:

   date > /dev/tty*nn*

   where *nn* is the number of the serial port you are using (for example, */dev/tty1a*).

For parallel printers:

date > /dev/lp*n*

where *n* is the number of the parallel port you are using (for example */dev/lp0*).

5. If you do not see the date printed on your printer, there is most likely some type of hardware malfunction, so verify the following:

For parallel printers:

- Make certain your cable is securely connected and all wires are good. Using the cable on a known good system and printing under DOS are good ways to test this.
- Recheck your printer configuration by verifying its switches in your printer hardware manual.
- Recheck the switches on your parallel card. It must also be recognized at bootup. You can verify this by running the **hwconfig**(C) command, or checking the file */usr/adm/messages* for a message similar to the following:

```
parallel 0x378-0x37A   07    -    unit=0
```

To confirm that your card is recognized, enter the following command:

hwconfig name=parallel

If the card is recognized, an entry will be printed that has similar information to the entry above.

For serial printers:

- Make certain you are using the non-modem control device, for example: */dev/tty1a*, not */dev/tty1A*. (For more information on the naming convention for serial ports, see **serial**(HW).) Try using a cable with only pins 2, 3, and 7 connected.
- Recheck your printer configuration by verifying its switches in your printer hardware manual.
- Recheck your switches on your serial port. If you are using a multiport card, try other lines on that card and be sure it does not conflict with the standard COM ports.

- Try attaching the printer to a standard serial port, COM1 or COM2, to see if the printer and cabling are correct.

6. Once you have the printer correctly hooked up and responding, you are ready to run **sysadmsh**. You must know the port you are using or the UNIX pathname of the device (for example, */dev/tty1a*) and the line printer interface program you are going to use. A model interface program is supplied with your UNIX system. For more information on printer interface programs, see "Printer Interface Programs" later in this chapter.

7. The **sysadmsh** "Printers" selection displays a form with a series of fields that you must fill in. If you make a mistake while responding to the questions, just press the **Esc** key and start again. To begin, return to the desktop and double-click on the Sysadmsh icon to bring up the main **sysadmsh**(ADM) menu. Then make the following selection:

   Printers→Configure→Add

8. The following form is displayed (the fields are discussed below):

```
                                                          Add
Enter ...

/                                    Thursday September 21, 1989 1:06

                          Adding a Printer

  Printer name            [              ]

  Comment                 [                                    ]

  Class name              [              ]

  Use printer interface   [Existing]  Copy   New
  Name of interface       [                                    ]

  Connection              [Direct]    Call-up
    Device name           [                   ]
    Dial-up information   [                   ]
  Device                  [Hardwired]   Login

  Require banner           Yes  [No]
```

Here is an explanation of each field:

Printer name Name of the new printer

Comment Comment which describes the printer

Class name Name of the class associated with this printer (**F3** for list)

Use printer interface Use an existing, copy or new printer interface

Name of interface Name of the interface (or **F3** to list existing interfaces)

Connection Whether the printer is directly connected to the system, or must be called up via a modem or network

Device Whether connection is dedicated to the printer, or is also used for a login terminal (will be disabled by scheduler)

Device name Name of device to which the printer is connected (for example, */dev/tty01* for a serial printer and */dev/lp0* for a parallel printer)

Dial-up information Modem phone number or network system name

Require banner Force a banner to always be printed or allow the user to request no banner be printed

When you finish filling in the form, it is executed and the new configuration is in place. To use the printer, you must also start the print service, enable the printer and allow the printer to accept requests. Do this by using the making the following **sysadmsh** selections:

Printers→Schedule→Begin
Printers→Schedule→Enable
Printers→Schedule→Accept

In the case of the **Enable** and **Accept** selections, you must supply the name of the printer when prompted to do so.

For more information on printer maintenance commands, see sections "Starting and Stopping the Print Service," "Managing the Primary Load," and "Enabling and Disabling Printers." The **sysadmsh** includes all these functions, supplementing the **lpadmin**(ADM) command.

Summary of User Commands

The print service has three regular user commands, which are shown in Table 12.1.

Table 12.1.

User Commands for the Print Service

Command	Description
cancel	Cancels a request for a file to be printed.
lp	Sends a file or files to a printer.
lpstat	Reports the status of the LP system.

In addition to sending requests to the print service system, check the status of requests, and canceling requests, users can be given the ability to disable and enable a printer. The idea is that if a user finds a printer is malfunctioning in some way, it should not be necessary to call the administrator to turn the printer off. On the other hand, it may not be reasonable in your printing environment to allow regular users to disable a printer. You can control whether other users have access to the two commands shown in Table 12.2 by assigning or revoking the **printerstat** authorization (see "Altering/Assigning User Subsystem Authorizations" in the "Administering User Accounts" chapter in this Guide).

Table 12.2.

Privileged User Commands for the Print Service

Command	Description
disable	Deactivates the named printer(s).
enable	Activates the named printer(s).

Summary of Administrative Commands

A separate set of commands available for the LP administrator is shown in Table 12.3. These commands are found in the */usr/lib* directory. If you expect to use them frequently, you might find it convenient to include that directory in your PATH variable. To use the administrative commands, you must be logged in as either *root* or have the *lp authorization* (see the "Administering User Accounts" chapter for an explanation of *authorizations.*)

Note that all these commands are accessible under the **sysadmsh** "Printers" selection. You'll also probably need to use the commands for disabling and enabling a printer and the rest of the commands described under the section "Summary of User Commands" above.

Table 12.3.

Administrative Commands for the LP Print Service

Command	Description
/usr/lib/accept	Permits job requests to be queued for a specified destination.
/usr/lib/reject	Prevents jobs from being queued for a specified destination. Described on the same manual page as **accept**(ADM).

/usr/lib/lpadmin	Sets up or changes printer configurations.
/usr/lib/lpforms	Sets up or changes preprinted forms. (Enter **/usr/lib/lpadmin** to mount a form.)
/usr/lib/lpmove	Moves output requests from one destination to another. Described on the same manual page as **lpsched**(ADM).
/usr/lib/lpsched	Starts the print service.
/usr/lib/lpshut	Stops the print service. Described on the same manual page as **lpsched**(ADM).
/usr/lib/lpusers	Sets or changes the default priority and priority limits the users of the print service can request.

These commands are also accessed through the **sysadmsh** "Printers" selection, which is much easier than the complex syntax of the LP commands.

Starting and Stopping the LP Print Service

Under normal operation, you should never have to start or stop the print service manually. It automatically starts each time the operating system starts, and stops each time the operating system stops. However, if you need to stop the print service without stopping the operating system as well, you can do so by following the procedure described in the next section.

Stopping the print service causes all printing to cease within seconds. Any print requests that have not finished printing are printed in their entirety after the print service restarts. The printer configurations, forms, and filters in effect when the print service is stopped are restored after it is restarted.

NOTE: To start and stop the print service manually, you must be logged in as either the super-user *root* or a user with the **lp** authorization.

Manually Stopping the Print Service

To stop the print service manually, enter the following command:

/usr/lib/lpshut

Δ **sysadmsh** users select: Printers→Schedule→Stop

This message is displayed:

```
Print services stopped
```

All printing ceases within a few seconds. If you try to stop the print service when it is not running, you see the message:

```
Print services already stopped
```

NOTE: Jobs can pass through a printer that is not online. If a printer is not online or operating properly, you should disable the printer.

Manually Starting the Print Service

To restart the print service manually, enter the following command:

/usr/lib/lpsched

Δ **sysadmsh** users select: Printers→Schedule→Begin

This message is displayed:

```
Print services started
```

It may take a minute or two for the printer configurations, forms, and filters to be re-established before any saved print requests start printing. If you try to restart the print service when it is already running, you see the message:

```
Print services already active
```

NOTE: You do not have to stop the print service to change printer configurations or to add forms or filters.

Canceling a Print Request

To cancel a printout you have requested, use the **cancel**(C) command. When you request a printout, the system displays a request ID for your job. For example, if you send a job to printer "laser" on your system, UNIX displays the request ID as:

```
request id is laser-number
```

where *number* is the number assigned to your job. To cancel the job before it begins printing, use the following command:

cancel laser-*number*

Δ **sysadmsh** users select: Printers→Request→Cancel

The printout is canceled.

Most systems print quickly, so a **cancel** command must be used promptly to have any effect.

Enabling and Disabling Printers

The **enable** command allows **lpsched** to print files on printers. A printer can accept requests for printing after the **accept** command is given for it, but to print the files, the **enable** command must be given as well.

For example, to enable a printer named "daisy," enter:

enable daisy

Δ **sysadmsh** users select: Printers→Schedule→Enable

You can disable printers with the **disable** command. The scheduler, **lpsched**, does not send printing requests to disabled printers regardless of their acceptance status. The **-r** option allows you to send a message to users explaining why a printer was disabled.

For example, to disable a printer named "laser" because of a paper jam, enter:

disable -r"paper jam" laser

Users requesting the status of "laser" with the command **lpstat -plaser** receive the following message:

```
printer laser disabled since Dec 5 10:15
paper jam
```

For more information on these two commands, see **enable**(C) and **disable**(C)

Adding a Printer to a Class

It is occasionally convenient to treat a collection of printers as a single class. The benefit is that a person can submit a file for printing by a member of a class, and the print service picks the first printer in the class that it finds free. This allows faster turnaround, as printers are kept as busy as possible.

Classes are not needed if the only purpose is to allow a user to submit a print request by type of printer. The **lp -T** *type* command lets a user submit a file and specify its type. The first available printer that can handle the type of file prints the file. One use of classes is to put into a class a series of printers that should be used in a particular order. If you have a high-speed printer and a low-speed printer, for instance, you probably want the high-speed printer to handle as many print requests as possible, with the low-speed printer reserved for use when the other is busy. Because the print service always checks for an available printer in the order the printers were added to a class, you could add the high-speed printer to the class before the low-speed printer and let the print service route print requests in the order you wanted.

Add a printer to a class using the following command:

/usr/lib/lpadmin -p *printername* **-c** *classname*

Δ **sysadmsh** users select: Printers→Configure→Modify

If the class *classname* does not exist yet, it is created.

NOTE: Class names and printer names must be unique. This allows a user to specify the destination for a print request without having to know whether it is a class of printers or a single printer. Thus, you can not have a class and printer with the same name.

Until you add a printer to a class, it does not belong to any.

Setting the System Default Destination

You can define the printer or class used to print a file when the user has not explicitly asked for a particular destination and has not set the LPDEST shell variable. The printer or class must already exist first.

Make a printer or class the default destination by entering the following command:

/usr/lib/lpadmin -d *printername* or *classname*

Δ **sysadmsh** users select: Printers→Configure→Default

If you later decide that there should be no default destination, enter a null *printername* or *classname* as in the following command:

/usr/lib/lpadmin -d

Δ **sysadmsh** users select: Printers→Configure→Default

If you do not set a default destination, there is none. Users must explicitly name a printer or class in each print request, or they have to set the LPDEST shell variable with the name of a destination.

For C-shell:

setenv LPDEST=*printer*

For Bourne shell:

LPDEST=*printer*; export LPDEST

Mounting a Form or Print Wheel

NOTE: See the "Forms" section in this chapter for information about preprinted forms.

Before the print service starts to print files that need a preprinted form or print wheel, you have to mount it on a printer. If alerting has been set on the form or print wheel, you are alerted when enough print requests are queued for it to be mounted.

When you mount a form, you may wish to see if it is lined up properly. If an alignment pattern is registered with the form, you can ask that this be repeatedly printed until you have adjusted the printer so that the alignment pattern looks correct.

Mounting a form or print wheel involves first loading it onto the printer and then telling the print service that it is mounted. Because it is difficult to do this on a printer that is currently printing and because the print service continues to print files not needing the form on the printer, you will probably have to disable the printer first. Thus, the proper procedure is as follows:

1. Disable the printer using the **disable** command.
2. Mount the new form or print wheel as described later in this section.
3. Re-enable the printer using the **enable** command. (The **disable** and **enable** commands were described earlier in the "Enabling and Disabling Printers" section of this chapter.)

After loading the new form or print wheel into the printer, enter the following command to tell the print service to mount it. (This command is shown on two lines for readability; it must be entered as one line.)

/usr/lib/lpadmin -p *printername* **-M -S** *print-wheelname*
-f *formname* **-a -o filebreak**

Δ **sysadmsh** users select: Printers→Auxiliary→PPforms→Configure

Leave out **-S***print-wheelname* if you are mounting just a form, or leave out the **-f***formname* **-a -o filebreak** if you are mounting just a print wheel.

If you are mounting a form, you are asked to press the **Return** key before each copy of the alignment pattern is printed. After the pattern is printed, you can adjust the printer and press the return key again. If no alignment pattern is registered, you are not asked to press the key. You can drop the **-a** and **-o filebreak** options if you do not want to bother with the alignment pattern.

The **-o filebreak** option tells the LP print service to add a *formfeed* after each copy of the alignment pattern. The actual control sequence used for the *formfeed* depends on the printer involved and is obtained from the **terminfo** database. If the alignment pattern already includes a formfeed, leave out the **-o filebreak** option.

If you want to unmount a form or print wheel, use the following command:

/usr/lib/lpadmin -p *printername* **-M -S none -f none**

Δ **sysadmsh** users select: Printers→Auxiliary→PPforms→Remove

Leave out **-S none** if you just want to unmount a form; likewise, leave out **-f none** if you just want to unmount a print wheel.

Until you mount a form on a printer, only print requests that do not require a form are sent to it. Likewise, until you mount a print wheel on a printer, only print requests that do not require a particular print wheel are sent to it.

Removing a Printer or Class

You can remove a printer or class if it has no pending print requests. If there are pending requests, you have to first move them to another printer or class using the **lpmove** command, or remove them using the **cancel** command.

Removing the last remaining printer of a class automatically removes the class as well. However, the removal of a class does not cause the removal of printers that were members of the class. If the printer or class removed is also the system default destination, the system no longer has a default destination.

To remove a printer or class, enter the following command:

/usr/lib/lpadmin -x *printername* or *classname*

Δ **sysadmsh** users select: Printers→Configure→Remove

If all you want to do is remove a printer from a class but not delete the printer, enter the following command:

/usr/lib/lpadmin -p *printername* **-r** *classname*

Δ **sysadmsh** users select: Printers→Configure→Modify

Managing the Printing Load

Occasionally, you may need to stop accepting print requests for a printer or move print requests from one printer to another. There are various reasons for doing this, such as the following:

- The printer needs periodic maintenance.
- The printer is broken.
- The printer was removed.
- You changed the configuration so that the printer can be used differently.
- Too many large print requests are queued for one printer and should be evenly distributed.

If you are going to make a big change in the way a printer is used, such as stopping its ability to handle a certain form, changing the print wheels available for it, or disallowing some people from using it, print requests that are currently queued for printing on it must be moved or canceled. The print service attempts to find alternate printers, but only if the user does not care which printer is used. Such requests are not automatically moved; if you do not move them first, the print service cancels them.

If you decide that a printer is to be taken out of service, its configuration is to be changed, or it is too heavily loaded, you can move print requests from it and reject additional requests for it. Use the **lpmove** and **reject** commands for this. If you do reject requests for a printer, you can later accept requests using the **accept** command.

Rejecting Requests for a Printer or Class

To stop accepting any new requests for a printer or a class of printers, enter the following command:

/usr/lib/reject -r "*reason*" *printername* or *classname*

Δ **sysadmsh** users select: Printers→Schedule→Reject

You can reject requests for several printers or classes in one command by listing their names on the same line, separating the names with spaces. The *reason* is displayed whenever anyone tries to print a file on the printer. You can drop it (and the **-r**) if you do not want to give a reason.

Although the **reject** command stops any new print requests from being accepted, it does not move or cancel any requests currently queued for the printer. These continue to print as long as the printer is enabled.

Accepting Requests for a Printer or Class

The **accept** command allows printers or classes of printers to accept print requests made with the **lp** command. You can allow a printer to accept requests after it has been properly configured.

After the condition that led to denying requests is corrected or changed, enter the following command to start accepting new requests:

/usr/lib/accept *printername* or *classname*

Δ **sysadmsh** users select: Printers→Schedule→Accept

Again, you can accept requests for several printers or classes in one command by listing their names on the same line. You will always have to use the **accept** command for a new printer or class after you have added it because the print service does not initially accept requests for new printers or classes.

Moving Requests to Another Printer

If you have to move requests from one printer or class to another, enter one of the following commands:

/usr/lib/lpmove *request-id printername*
/usr/lib/lpmove $printername_1$ $printername_2$

Δ **sysadmsh** users select: Printers→Request→Move

You can give more than one request ID before the printer name in the first command. The first command moves the listed requests to the named printer. The latter command moves all requests currently queued for the first printer to the second printer. When the latter command is used, the print service also no longer accepts requests for the first printer (this has the same effect as the **reject** command).

Managing Queue Priorities

The print service provides a simple priority mechanism that people can use to adjust the position of a print request in the queue. Each print request can be given a priority level by the person who submits it; this is a number from 0 to 39, with *smaller* numbers indicating *higher* levels of priority. Requests with higher priority (smaller numbers) are placed ahead of requests with lower priority (larger numbers).

In this way, if you decide that your print request is of too low a priority, you can set a higher priority (lower value) when you submit the file for printing. If you decide that your print request is of too high a priority, you can set a lower priority (higher value) when you submit the file for printing.

A priority scheme this simple does not work if there are no controls on how high one can set the priority. You can define the following characteristics of this scheme:

- Each user can be assigned a priority limit. One cannot submit a print request with a priority higher than his or her limit, although one can submit a request with a lower priority.
- A default priority limit can be assigned for the balance of users not assigned a personal limit.
- A default priority can be set. This is the priority given print requests to which the user does not assign a priority.

By setting the characteristics according to your needs, you can prevent lower priority printing tasks (such as regular printing by most staff members) from interfering with higher priority printing tasks (such as payroll check printing by the accounting staff).

You may find that you want a critical print request to print ahead of any others, perhaps even if it has to preempt the currently printing request. You can have the print service give *immediate* handling to a print request and put on *hold* another print request. This lets the urgent print request print and delays the current print request until you have it *resumed.*

The **lpusers** command lets you assign both priority limits for users and priority defaults. In addition, you can use the **lp -i** *request-id* **-H hold** and **lp -i** *request-id* **-H immediate** commands to put a request on hold or move it up for immediate printing, respectively. These commands are discussed in detail next.

Setting Priority Limits

To set a user's priority limit, enter the following command:

/usr/lib/lpusers -q *priority-level* **-u** *username*

You can set the limit for a group of users by listing their names after the **-u** option. Separate multiple names with a comma or space (enclose the list in quotes if you use a space, though). The *priority-level* is a number from 0 to 39. As mentioned before, the lower the number, the higher the priority, or, in this case, the priority limit.

If you want to set the priority limit for all other users, enter the following command:

/usr/lib/lpusers -q *priority-level*

Δ **sysadmsh** users select: Printers→Priorities→Default

This sets the default limit; the default applies to those people who have not been given a personal limit using the earlier **lpusers** command.
If you later decide that someone should have a different priority limit, just re-enter the first command above with a new limit. If you decide that someone with a personal limit should have just whatever the default limit is, enter the following command:

/usr/lib/lpusers -u *username*

Δ **sysadmsh** users select: Printers→Priorities→Remove

Again, you can do this for more than one person at a time by giving a list of names. Using the **lpusers** command with just the **-u** option puts the users in the *default limit* category.

If you do not set a default limit, people without personal limits are limited to priorities in the range of 20 to 39.

Setting a Default Priority

You can set the default priority that should be assigned to those print requests submitted without a priority. Use the following command:

/usr/lib/lpusers -d *priority-level*

Δ **sysadmsh** users select: Printers→Priorities→Highest

Do not confuse this default with the *default limit*. This default is applied when a user does not give a priority; the *default limit* is applied if you have not assigned a limit for a user—it is used to limit the user from giving too high a priority.

NOTE: If the default priority is greater than the limit for a user, the limit is used instead.

If you do not set a default priority, the print service uses the default of 20.

Examining the Priority Limits and Defaults

You can examine all the settings you have assigned for priority limits and defaults by entering the following command:

/usr/lib/lpusers -l

Δ **sysadmsh** users select: Printers→Priorities→List

Moving a Request Around in the Queue

Once a user has submitted a print request, you can move it around in the queue to some degree. For example, you can:

- adjust the priority to any level regardless of the limit for the user,
- put it on hold and let other requests print ahead of it,
- put it at the head of the queue for immediate printing.

You use the regular **lp** user command to do each of these.

Changing the Priority for a Request

Print requests that are still waiting to print can be reassigned a new priority. This repositions it in the queue, putting it ahead of lower priority requests but behind any others at the same or higher priority. The priority limit assigned to the user (or the default priority limit) has no effect because you override this limit as the administrator.

Enter the following command to change the priority of a request:

lp -i *request id* **-q** *new-priority-level*

You can change only one request at a time with this command. If a request is already printing, you cannot change its priority.

Putting a Request on Hold

Any request that has not finished printing can be put on hold. You can stop its printing, if it currently is printing, and keep it from printing until you resume it. Another user, however, cannot resume a print request that you put on hold.

Enter the following command to place a request on hold:

lp -i *request-id* **-H hold**

Enter the following command to resume the request:

lp -i *request-id* **-H resume**

Once resumed, a request continues to move up the queue and will prints. If it had been printing when you held it, it is the next request to print. Normally it starts printing from the beginning, with page one, but you can have it start printing at a later page. Enter the following command to resume the request at a different page:

lp -i *request-id* **-H resume -P** *starting-page-*

The final dash is needed to specify the starting page and all subsequent pages.

NOTE: The ability to print a subset of pages requires the presence of a filter that can handle this. The default filter used by the print service cannot handle it. An attempt to resume a request on a later page is rejected if an appropriate filter is not being used.

Moving a Request to the Head of the Queue

You can move a print request to the head of the queue, where it is the next job eligible for printing. If it must start printing immediately, but another request is currently printing, you can hold the other request as described above.

Enter the following command to move a print request to the head of the queue:

lp -i *request-id* **-H immediate**

Only the system administrator can move a request like this; regular users cannot use the **-H immediate** option.

NOTE: If you set more than one request for immediate printing, they print in the reverse order set; that is, the request moved to the head of the queue most recently prints first.

Examining a Printer Configuration

Once you define a printer configuration, you probably want to review it to see if it is correct. If after examining the configuration you find you made a mistake, just re-enter the command that applies to the part that is wrong.

Use the **lpstat** command to examine both the configuration and the current status of a printer. A short form of this command gives just the status; you can use it to see if the printer exists and if it is busy, idle, or disabled. A long form of the command adds the complete configuration.

Enter one of the following commands to examine a printer:

lpstat -p *printername*
lpstat -p *printername* **-l**

The second command is the long form. With either command you should see something like the following:

```
printer printer-name now printing request-id. enabled
since date.

printer printer-name is idle. enabled since date.

printer printer-name disabled since date.
        reason

printer printer-name waiting for auto-retry.
        reason
```


The "waiting for auto-retry" output shows that the LP print service failed in trying to use the printer (because of the *reason* shown) and that the print service will try again later.

With the long form of the command, you should also see the following items on the output:

```
Form mounted: form-name
Content types: content-type-list
Printer type: printer-type
Description: comment
Connection: connection-info
Interface: path-name
On fault: alert-method
After fault: fault-recovery
Users allowed:
        user-list
Forms allowed:
        form-list
Banner required
Character sets:
        character-set-list
Default pitch: integer CPI, integer LPI
Default page size: scaled-decimal-number wide,
       scaled-decimal-number long
Default port settings: stty-option-list
```

See "Enabling and Disabling Printers" earlier in this chapter for information.

Troubleshooting the Print System

If you are having difficulty getting your printer to work, here are a few suggestions for you to try.

No Output Nothing Prints

The printer is sitting idle; nothing happens. First, check the documentation that came with the printer to see if there is a self-test feature you can invoke; make sure the printer is working before continuing.

Is the Printer Connected to the Computer?

Check to make sure that the printer is connected to the printer. Refer to your printer's owner's manual for installation instructions.

Is the Printer Enabled?

The printer must be *enabled* in two ways. First, the printer must be turned on and ready to receive data from the computer. Second, the print service must be ready to use the printer. Set up the printer as described in the "Installing a Printer" section of this chapter. If you receive error messages when doing this, follow the *fixes* suggested in the messages. When you finish setting up the printer, use the following commands:

/usr/lib/accept *printername*
enable *printername*

Δ **sysadmsh** users select: Printers→Schedule→Accept
Printers→Schedule→Enable

where *printername* is the name you assigned to the printer for the print service. Then submit a sample file (like */etc/passwd*) for printing:

lp -d *printername* **-T** *printer-type filename*

Δ **sysadmsh** users select: Dirs/Files→Print

If you did not give a printer type for the printer, leave out the **-T** *printer-type* option.

Is the Baud Rate Correct?

If the baud rate (the rate at which the computer sends data to the printer) does not match with the printer, sometimes nothing prints. See "Illegible Output."

Illegible Output

The printer tries printing, but it is not what you expected and certainly is not readable.

Is the Baud Rate Correct?

Usually when the baud rate does not match with the printer, you get some output but it does not look at all like what you submitted for printing. Random characters appear with an unusual mix of special characters and unlikely spacing.

Read the documentation that came with the printer to find out what its baud rate is. It should probably be set at 9600 baud for optimum performance, If it is not set to 9600 baud, you can have the print service use the correct baud rate (by default it uses 9600). If the printer is connected via a parallel port, the baud rate does not matter.

To set a different baud rate for the print service to use, use the following command:

/usr/lib/lpadmin -p *printername* **-o stty=***baud-rate*

Δ **sysadmsh** users select: Printers→Configure→Parameters

The "Default stty" field is in the third part of the form; enter the baud rate number. Now submit a sample file for printing (explained earlier in "Mounting a Form or Print Wheel").

Is the Parity Setting Correct?

Some printers use a *parity bit* to ensure that the data they receive for printing has not been garbled in transmission. The parity bit can be encoded in a few different ways, and both the computer and the printer must agree on which to use. If they do not agree, some characters are not printed or have another character substituted. Generally, though, the output looks approximately correct, with the spacing of "words" typical for your document and many letters in their correct place.

Check the documentation for the printer to see what it expects. If your printer is directly connected to the computer with a fairly short wire (50 feet or so), it does not have to use the parity bit. The print service does not expect to set the parity bit by default. You can change this, however, by using the following **sysadmsh** selection:

/usr/lib/lpadmin -p *printername* **-o stty=oddp**
/usr/lib/lpadmin -p *printername* **-o stty=evenp**
/usr/lib/lpadmin -p *printername* **-o stty=-parity**

Δ **sysadmsh** users select: Printers→Configure→Parameters

And make one of the following additions to the "Default stty" field in part three of the form: **oddp**, **evenp**, **-parity**. The first sets odd parity generation, the second sets even parity. The last command sets the default, no parity. Select the option that matches what your printer needs.

Tabs Set Correctly?

If the printer does not expect to receive tab characters, the output may be there but all of it is jammed up against the right margin. See "No Left Margin/Run-on Text" later in this chapter.

Correct Printer Type?

See explanation under "Wrong Character Set or Font" later in this chapter.

Legible Printing, Wrong Spacing

The output is all there, it is readable, but it is double spaced, has no left margin, runs together, or zigzags down the page. These problems can be fixed by adjusting the printer settings (if possible) or having the print service use matching settings.

Double Spaced

To correct the double-spaced text, use either the **-onlcr** or **-tabs** option.

No Left Margin/Run-on Text

If there is no left margin and the text runs together, then use the **-tabs** option.

Zig Zags down the Page

If the output zig zags down your page, use the **onlcr** option. This is set by default, but you may have cleared it accidentally.

Correct Printer Type?

See next section, "Wrong Character Set of Font."

Wrong Character Set or Font

If the wrong printer type was selected when you set up the printer with the print service, the wrong control characters can be sent to the printer. The results are unpredictable and can cause output to disappear or be illegible, making it look like a problem described previously. A simpler problem to solve is when it sets the wrong character set or font.

If you do not know what printer type to give, try the following to examine the available printer types. First, if you think the printer type has a certain name, try the following command:

TERM=*printer-type* **tput longname**

The output of this command will appear on your terminal and is a short description of the printer identified by the *printer-type*. Try the names you think might be right until you find one that identifies your printer.

If you do not know what names to try, you can examine the */usr/lib/terminfo* directory to see what names are available. Note that there are probably many names in that directory. Enter the following command to examine the directory:

ls -R /usr/lib/terminfo | more

Pick names from the list that match one word or number identifying your printer. For example, the name 495 identifies the AT&T 495 Printer. Try each of the names in the other command above.

When you have the name of a printer type that you think is correct, set it in the print service by entering the following command:

/usr/lib/lpadmin -p *printername* **-T** *printer-type*

Δ **sysadmsh** users select: Printers→Configure→Parameters

Idle Printers

There are several reasons why you may find a printer idle and enabled but with print requests still queued:

- The printer has a fault. Automatic continuation of printing after a fault has been detected does not happen immediately. The print service waits about five minutes before trying again and keeps trying until a request prints successfully. You can force a retry immediately by enabling the printer:

  enable *printername*

  Δ **sysadmsh** users select: Printers→Schedule→Enable

- Lost *child process*. If the process controlling the printer is killed (by the system during periods of extremely heavy load or by an administrator), the print service may not realize it for a few minutes. Disabling the printer and then re-enabling it again forces the print service to check for the controlling process and restart one. Make sure the printer is really idle, though, because disabling a printer stops it in the middle of printing a request. Though the request is not lost, it does have to be reprinted in its entirety.

  disable *printername*
  enable *printername*

  Δ **sysadmsh** users select: Printers→Schedule→Disable
  Printers→Schedule→Enable

Customizing the Print Service

Although the print service tries to be flexible enough to handle most printers and printing needs, it cannot be complete. You may buy a printer that does not quite fit into the way the print service handles printers or may have a printing need that the standard features of the print service do not accommodate.

You can customize the print service in a few ways. This section tells you how you can:

- adjust the printer port characteristics
- adjust the **terminfo** database
- write an interface program

The diagram in Figure 12-1 gives an overview of the processing of a print request.

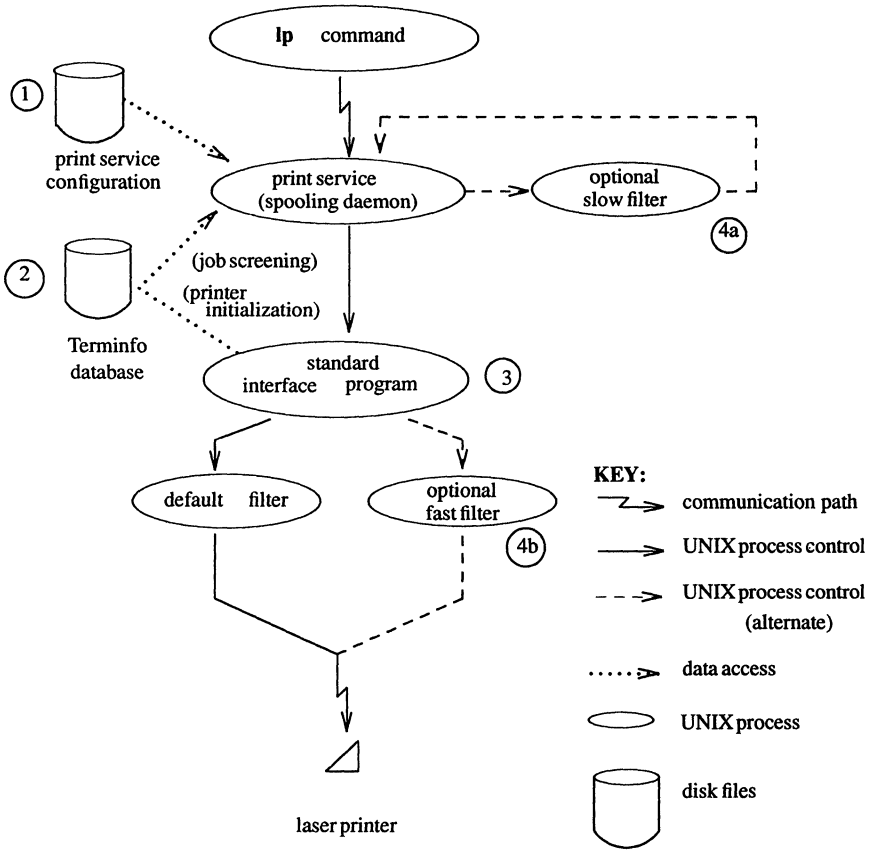


Figure 12-1. How LP processes print request lp -d laser file

Each print request is sent to a *spooling daemon* that keeps track of all the requests. The daemon is created when you start the LP print service. This UNIX system process is also responsible for keeping track of the status of the printers and slow filters; when a printer finishes printing a user's file, the daemon starts it printing another request, if one is queued.

You can customize the print service by adjusting or replacing some of the pieces shown in Figure 12-1. (The numbers are keyed to the diagram.)

1. For most printers, you need only change the printer configuration stored on disk. The earlier sections of this chapter have explained how to do this. Some of the more printer-dependent configuration data are the printer port characteristics: baud rate, parity, and so on.

2. For printers that are not represented in the **terminfo** database, you can add a new entry that describes the capabilities of the printer. This database is used in two parallel capacities: screening print requests to ensure that those accepted can be handled by the desired printer and setting the printer so it is ready to print the request.

   For instance, if the **terminfo** database does not show a printer capable of setting a page length requested by a user, the spooling daemon rejects the request. On the other hand, if it does show it capable, then the same information is used by the interface program to initialize the printer.

3. For particularly difficult printers or if you want to add features not provided by the delivered LP print service, you can change the standard interface program. This program is responsible for managing the printer: it prints the banner page, initializes the printer, and invokes a filter to send copies of the user's files to the printer.

4a,b. To provide a link between the applications used on your system and the printers, you can add slow and fast filters. Each type of filter can convert a file into another form, mapping one set of escape sequences into another, for instance, and can provide special setup by interpreting print modes requested by a user. Slow filters are run separately by the daemon to avoid tying up a printer. Fast filters are run so their output goes directly to the printer; thus, they can exert control over the printer.

Adjusting the Printer Port Characteristics

You should make sure that the printer port characteristics set by the print service match the printer communication settings. The standard printer port settings were designed to work with typical UNIX files and many printers, but they do not work with all files and printers. This is not really a customizing step, because a standard feature of the print service is to allow you to specify the port settings for each printer. However, it is an important step in getting your printer to work with the print service, so it is described in more detail here.

When you add a new printer, read the documentation that comes with it so that you understand what it expects from the host (the print service). Then read the manual page for the **stty**(C) command It summarizes the various characteristics that can be set on a terminal or printer port.

Only some of the characteristics listed in the **stty**(C) manual page are important for printers. The ones likely to be of interest to you are listed in the following table (but you should still consult the **stty**(C) manual page for others).

Printers connected directly to computers and those connected over some networks require that the printer port characteristics be set by the interface program. These characteristics define the low-level communications with the printer. Included are the baud rate; use of XON/XOFF flow control; 7, 8, or other bits per byte; style of parity; and output post-processing. The standard interface program uses the **stty** command to initialize the printer port, minimally setting the baud rate and a few other default characteristics.

The default characteristics applied by the standard interface program are listed below.

Table 12.4.

Default stty Options

Default	Meaning
9600	9600 baud rate
cs8	8-bit bytes
-cstopb	1 stop bit per byte
-parenb	no parity generation
ixon	enable XON/XOFF flow control
-ixany	allow only XON to restart output
opost	post-process data stream as listed below
-oluc	do not map lowercase to uppercase
onlcr	map linefeed into carriage-return/linefeed
-ocrnl	do not map carriage-return into linefeed
-nocr	output carriage-returns even at column 0
nl0	no delay after linefeeds
cr0	no delay after carriage-returns
tab0	no delay after tabs
bs0	no delay after backspaces
vt0	no delay after vertical tabs
ff0	no delay after formfeeds

You may find that the default characteristics are sufficient for your printers. However, printers vary enough that you are likely to find that you have to set different characteristics. See the **stty**(C) man page to find the complete list of characteristics.

If you have a printer that requires printer port characteristics other than those handled by the **stty** program, you must customize the interface program.

When you add a new printer, you can specify an additional list of port characteristics that should be applied when printing each user's file. The list you give will be applied after the default list so that you do not need to include in your list default items that you don't want to change. Specify the additional list as follows:

/usr/lib/lpadmin -p *printer-name* **-o "stty='***stty-option-list***'"**

Δ **sysadmsh** users select: Printers→Configure→Parameters

Note that both the double quotes and single quotes are needed if you give more than one item in the *stty-option-list*. If you do not include alternate printer port characteristics, the default list in the table will be used.

As one example, suppose your printer is to be used for printing graphical data, where linefeed characters should be output alone without an added carriage-return. You would enter the following command:

/usr/lib/lpadmin -p *printer-name* **-o "stty=-onlcr"**

Note that the single quotes are omitted because there's only one item in the list.

As another example, suppose your printer requires odd parity for data sent to it. You would enter the following command:

/usr/lib/lpadmin -p *printer-name* **-o "stty='parenb parodd cs7'"**

Adjusting the terminfo Database

The print service relies on a standard interface and the **terminfo** database to initialize each printer and set up a selected page size, character pitch, line pitch, and character set. Thus, it is usually sufficient to have the correct entry in the **terminfo** database to add a new printer to the print service. Several entries for popular printers are delivered in **terminfo** database entries with the print service package.

Each printer is identified in the **terminfo** database with a short name; this kind of name is identical to the kind of name used to set the **TERM** shell variable. For instance, the AT&T model 455 printer is identified by the name **455**.

If you cannot find a **terminfo** entry for your printer, you should add one. If you do not, you may still be able to use the printer with the print service, but you cannot get automatic selection of page size, pitch, and character sets, and you may have trouble keeping the printer set in the correct modes for each print request. Another option to follow instead of updating the **terminfo** entry is to customize the interface program used with the printer. See the next section for details on how to do this.

There are hundreds of items that can be defined for each terminal or printer in the **terminfo** database. However, the print service uses less than fifty of these, and most printers need even less than that. Table 12.5 lists the items that need to be defined (as appropriate for the printer) to add a new printer to the print service.

Table 12.5.
terminfo Definitions

terminfo item	Meaning
Booleans:	
daisy	Printer needs operator to change character set.
Numbers:	
bufsz	Number of bytes buffered before printing.
cols	Number of columns in a line.
it	Tabs initially every # spaces.
lines	Number of lines on a page.
orc	Horizontal resolution in units per character.
orhi	Horizontal resolution in units per inch.
orl	Vertical resolution in units per line.

(Continued on next page.)

Table 12.5.

terminfo Definitions (*Continued*)

terminfo item	Meaning
orvi	Vertical resolution in units per inch.
cps	Average print rate in characters per second.
Strings:	
cr	Carriage return.
cpi	Change number of characters per inch.
lpi	Change number of lines per inch.
chr	Change horizontal resolution.
cvr	Change vertical resolution.
csnm	List of character set names.
mgc	Clear all margins (top, bottom and sides).
hpa	Horizontal position absolute.
cud1	Down one line.
cuf1	Carriage right.
swidm	Enable double wide printing.
rwidm	Disable double wide printing.
ff	Page eject.
is1	Printer initialization string.
is2	Printer initialization string.
is3	Printer initialization string.
if	Name of initialization file.
iprog	Pathname of initializing program.
cud	Move carriage down # lines.
cuf	Move carriage right # columns.
rep	Repeat a character # times.
vpa	Vertical position absolute.

(*Continued on next page.*)

Table 12.5.

terminfo Definitions (*Continued*)

terminfo item	Meaning
scs	Select character set.
smgb	Set bottom margin at current line.
smgbp	Set bottom margin.
smgl	Set left margin at current column.
smglp	Set left margin.
smgr	Set right margin at current column.
smgrp	Set right margin.
smgt	Set top margin at current line.
smgtp	Set top margin.
scsd	Start definition of a character set.
ht	Tab to next 8-space tab stop.

Consult the manual page for the **terminfo**(M) file structure for details on how to construct a **terminfo** database entry for a new printer.

Once you make the new entry, you need to compile it into the database using the **tic** program. Just enter the following command:

tic *filename*

filename is the name of the file containing the **terminfo** entry you have crafted for the new printer.

NOTE: The LP print service gains much efficiency by *caching* information from the **terminfo** database. If you add or delete **terminfo** entries or change the values that govern pitch settings, page width and length, or character sets, you should stop and restart the LP print service so it can read the new information.

How to Write an Interface Program

NOTE: If you have an interface program that you have used with the LP Spooler Utilities before UNIX System V Release 3.2, it should still work with the print service. Note, though, that several **-o** options have been standardized and are passed to every interface program. These may interfere with similarly named options your interface program uses.

If you have a printer that is not supported by simply adding an entry to the **terminfo** database, or if you have printing needs that are not supported by the standard interface program, you can furnish your own interface program. It is a good idea to start with the standard interface program and change it to fit, rather than starting from scratch. You can find a copy of it under the name */usr/spool/lp/model/standard*.

What Does an Interface Program Do?

Any interface program performs the following tasks:

- Initializes the printer port, if needed. The generic interface program uses the **stty** command to do this.
- Initializes the physical printer. The generic interface program uses the **terminfo** and the **TERM** shell variable to get the control sequences to do this.
- Prints a banner page, if needed.
- Prints the correct number of copies of the request content.

An interface program is not responsible for opening the printer port. This is done by the print service, which calls a *dial-up* printer if that is how the printer is connected. The printer port connection is given to the interface program as standard output, and the printer is set to be the *controlling terminal* for the interface program so that a *hang-up* of the port causes a **SIGHUP** signal to be sent to the interface program.

A customized interface program must not terminate the connection to the printer or in any fashion uninitialize the printer. This allows the print service to use the interface program only for preparing the printer and printer port, while the printing of content is done elsewhere, by the print service, for example, for preprinted form alignment patterns.

How Is an Interface Program Used?

When the print service routes an output request to a printer, the interface program for the printer is invoked as follows:

/usr/spool/lp/admins/lp/interface/*P id user title copies options file1 file2 ...*

Arguments for the interface program are:

P	Printer name.
id	Request id returned by lp.
user	Login name of user who made the request.
title	Optional title specified by the user.
copies	Number of copies requested by user.
options	List of options separated by blanks, specified by user or set by the print service.
file	Full pathname of a file to be printed.

When the interface program is invoked, its standard input comes from */dev/null*, its standard output is directed to the printer port, and its standard error output is directed to a file that will be given to the user who submitted the print request.

The standard interface recognizes the following values in the list in *options*:

nobanner This option is used to skip the printing of a banner page; without it, a banner page is printed.

nofilebreak This option is used to skip page breaks between separate data files; without it, a page break is made between each file in the content of a print request.

cpi=*decimal-number*$_1$

lpi=*decimal-number*$_2$

These options say to print with *decimal-number*$_1$ columns per inch and *decimal-number*$_2$ lines per inch, respectively. The standard interface program extracts from the **terminfo** database the control sequences needed to initialize the printer to handle the character and line pitches.

The words *pica*, *elite*, and *compressed* are acceptable replacements for the *decimal-number*$_1$ and are synonyms for *10* columns per inch, *12* columns per inch, and as many columns per inch as possible.

length=*decimal-number*$_1$

width=*decimal-number*$_2$

These options specify the length and width, respectively, of the pages to be printed. The standard interface program extracts from the **terminfo** database the control sequences needed to initialize the printer to handle the page length and page width.

stty='*stty-option-list*'

The *stty-option-list* is applied after a default *list* as arguments to the **stty** command. The default list is used to establish a default port configuration; the additional list given to the interface program is used to change the configuration as needed.

The above options are either specified by the user when issuing a print request or by the print service from defaults given by the administrator for the printer (**cpi, lpi, length, width, stty**) or for the preprinted form used in the request (**cpi, lpi, length, width**).

Additional printer configuration information is passed to the interface program in shell variables:

TERM=*printer-type*

This shell variable specifies the type of printer. The value is used as a key for getting printer capability information from the extended **terminfo** database.

FILTER='*pipeline*'

This shell variable specifies the filter to use to send the request content to the printer; the filter is given control of the printer.

CHARSET=*character-set*

This shell variable specifies the character set to be used when printing the content of a print request. The standard interface program extracts from the **terminfo** database the control sequences needed to select the character set.

A customized interface program should either ignore these options and shell variables or should recognize them and treat them in a consistent manner.

Customizing the Interface Program

You want to make sure that the custom interface program sets the proper **stty** modes (terminal characteristics such as baud rate or output options). The standard interface program does this, and you can follow suit. Look for the section that begins with the shell comment:

```
## Initialize the printer port
```

Follow the code used in the standard interface program. It sets both the default modes and the adjusted modes given by the print service or the user with a line like the following:

stty *mode options* **0<&1**

This command line takes the standard input for the **stty** command from the printer port. An example of an **stty** command line that sets the baud rate at 1200 and sets some of the option modes is shown here:

stty -parenb -parodd 1200 cs8 cread clocal ixon 0<&1

One printer port characteristic not set by the standard interface program is hardware flow control. The way that this is set will vary depending on your computer hardware. The code for the standard interface program suggests where this and other printer port characteristics can be set. Look for the section that begins with the shell comment

```
# Here you may want to add other port initialization code.
```

Because different printers have different numbers of columns, make sure the header and trailer for your interface program correspond to your printer. The standard interface program prints a banner that fits on an 80-column page (except for the user's title which may be longer). Look for the section in the code for the standard interface program that begins with the shell comment

```
## Print the banner page
```

The custom interface program should print all user-related error messages on the standard output or on the standard error. The messages sent to the standard error is mailed to the user; the messages printed on the standard output end up on the printed page, where they can be read by the user when he or she picks up the output.

When printing is complete, your interface program should exit with a code that tells the status of the print job. Exit codes are interpreted by the print service as shown in Table 12.6.

Table 12.6.
Exit Codes

Code	Meaning to the Print Service
0	The print request completed successfully. If a printer fault occurred, it was cleared.
1 to 127	A problem was encountered in printing this particular request (for example, too many non-printable characters or the request exceeds the printer capabilities). This problem does not affect future print requests. The print service notifies the person who submitted the request that there was an error in printing it. If a printer fault occurred, it was cleared.
128	Reserved for internal use by the LP print service. Interface programs must not exit with this code.
129	A printer fault was encountered in printing the request. This problem affects future print requests. If the fault recovery for the printer directs the print service to wait for the administrator to fix the problem, it disables the printer. If the fault recovery is to continue printing, the print service does not disable the printer but tries printing again in a few minutes.
> 129	These codes are reserved for internal use by the print service. Interface programs must not exit with codes in this range.

As the table shows, one way of alerting the administrator to a printer fault is to exit with a code of 129. Unfortunately, if the interface program exits, the print service has no choice but to reprint the request from the beginning when the fault is cleared. Another way of getting an alert to the administrator but without requiring reprinting the entire request, is to have the interface program send a fault message to the print service but wait for the fault to clear. When the fault clears, the interface program can resume printing the user's file. When done printing, it can give a zero exit code just as if the fault never occurred. An added advantage is that the interface program can detect when the fault is cleared automatically so that the administrator does not have to enable the printer.

Fault messages can be sent to the print service using the **lp.tell** program. This is referenced using the $LPTELL shell variable in the standard interface code. The program takes its standard input and sends it to the print service, where it is put into the message that alerts the administrator to the printer fault. If its standard input is empty, **lp.tell** does not initiate an alert. Examine the standard

interface code immediately after these comments for an example of how the **lp.tell** ($LPTELL) program is used:

```
# Here's where we set up the $LPTELL program to capture
# fault messages.

# Here's where we print the file.
```

With the special exit code 129 or the **lp.tell** program, there is no longer the need for the interface program to disable the printer itself. Your interface program can disable the printer directly, but doing so overrides the fault alerting mechanism. Alerts are sent only if the print service detects the printer has faulted, and the special exit code and the **lp.tell** program are its main detection tools.

If the print service has to interrupt the printing of a file at any time, it kills the interface program with a signal 15 (see **kill**(C) and **signal**(S)). If the interface program dies from receipt of any other signal, the print service assumes that future print requests are not affected and continues to use the printer. The print service notifies the person who submitted the request that it did not finish successfully.

The signals SIGHUP, SIGINT, SIGQUIT, and SIGPIPE (trap numbers 1, 2, 3, and 13) start out being ignored when the interface is invoked. The standard interface changes this to trap these signals at appropriate times. The standard interface considers receipt of these signals as meaning the printer has a problem and issues a fault. This is the program the print service uses to manage the printer each time a file is printed. It has four main tasks:

- to initialize the printer port (the connection between the computer and the printer),
- to initialize the printer (restore it to a normal state in case a previously printed file has left it in an unusual state) and set the character pitch, line pitch, page size, and character set requested by the user,
- to print a banner page, and
- to run a filter to print the file.

How to Add an Interface Program

If you do not choose an interface program, the standard one provided with the print service is used. This should be sufficient for most of your printing needs. If you prefer, however, you can change it to suit your needs or completely rewrite your own interface program, and then specify it when you add a new printer.

If you plan to use the standard interface program, you need not specify it when adding a printer. However, if you use a different interface program, you can either refer to it by its full pathname or by another printer using the same interface program.

To identify a customized interface program by name, give the printer name and the pathname of the interface program as follows:

/usr/lib/lpadmin -p *printername* **-i** *pathname*

To identify a customized interface program by reference to another printer, give the printer names as follows:

/usr/lib/lpadmin -p *printname*$_1$ **-e** *printname*$_2$

printname$_1$ should be replaced with the name of the printer you are adding; *printname*$_2$ should be replaced with the name of the printer already added that is using the customized interface program.

To identify an interface program by reference to a model interface program, give the printer name and model name as follows:

/usr/lib/lpadmin -p *printername* **-m** *modelname*

Specialized Configuration Options

Although the default values for printer configuration are usually sufficient for most needs, there are a number of options to configure individual aspects of printer operations. This includes such options as fault alerting and recovery. The following is a list of additional information that can be given to define the configuration of each printer:

- printer type
- content types
- character sets or print wheels
- fault alerting
- fault recovery
- default printing attributes

You need to give very little of this information to add a new printer to the print service; however, the more information you provide, the better the printer is managed for you and the better it can serve the people using the print service.

The descriptions in the following sections help you understand what this printer configuration information means and how it is used so that you can decide how to configure your printers. In each section, you are also shown how to specify this information when adding a printer. While you can follow each of the sections in order and correctly configure a printer in several steps, you may want to wait until you have read all of the sections before adding a printer so that you can do it in one step.

Printer Type

The printer type is important for the proper use of the printer. The print service uses the printer type to extract information about the printer from the **terminfo** database. This information describes the capabilities of the printer so that you can be warned if some of the configuration information you provide is not appropriate for the printer. The information also describes the control data to use to initialize the printer before printing a file. While you are not required to specify a printer type, you are urged to specify one so that better print services are provided.

The printer type is the generic name for the printer. Specify the printer type as follows:

/usr/lib/lpadmin -p *printername* **-T** *printer-type*

Δ **sysadmsh** users select: Printers→Configure→Parameters

If you do not define the printer type, the default **unknown** is used. This produces empty results when the print service looks up information about the printer, so the print service cannot verify certain requests or initialize the printer.

Content Types

While the printer type information tells the print service what type of printer is being added, the content type information tells the print service what types of files can be printed. Most printers can print only one type of file; for them, the content type is likely to be identical to the printer type. Some printers, though, can accept several different types of files and print their contents properly. When adding this kind of printer, you should list the names of the content types it accepts.

When a file is submitted to the print service for printing, the print service searches for a printer capable of handling the job. The print service can identify an appropriate printer through either the content-type name or the printer-type name. Therefore, you can specify either name (or no name) when submitting a file for printing.

Content-type names may look a lot like printer-type names, but you are free to choose names that mean something to you and the people using the printer. (The names **simple**, **terminfo**, or **any** are recognized as having particular meanings by the print service; be sure to use them consistently.)

The names must contain no more than 14 characters and may include only letters, digits, and underscores. If the same content type is printable by several different types of printers, you should use the same content type names when you add those printers. This makes it easier for the people using the printers because they can use the same name to identify the type of file they want printed regardless of the printing destination.

For example, several manufacturers produce printers that accept PostScript files. While these printers may need different printer types so that each can be properly initialized (assuming the initialization control sequences are different), they may all be capable of handling the same type of input file, which you call, perhaps, **postscript**. As another example, several manufacturers produce printers that accept ANSI X3.64 defined escape sequences. However, the printers may not support all the ANSI capabilities or may support different sets of capabilities. You may want to give different content-type names for these printers to differentiate them.

You do not have to list the content types for a printer. If you do not, the printer type is used as the name of the content type the printer can handle. If you have not specified a printer type, the print service assumes the printer can print only files of content type **simple**. This may be sufficient if you require people to pick the proper printer and make sure the files are properly prepared for the printer before they are submitted for printing.

The most common type of file on the UNIX system is known as **simple**. This file is assumed to contain just printable ASCII characters and the following control characters:

backspace	Moves the carriage back one space. except at the beginning of a line
tab	Moves the carriage to the next tab stop. which is normally every 8 columns on most printers
linefeed	Moves the carriage to the beginning of the next line (may require special port settings for some printers—see the next section "Printer Port Characteristics.")
form feed	Moves the carriage to the beginning of the next page.
carriage return	Moves the carriage to the beginning of the same line (may fail on some printers).

The word "carriage" may be archaic for modern laser printers, but similar actions apply. If a printer can handle a **simple** type of file, you should include it in the content type list when you add the printer and specify the content type(s) the printer can handle. If you do not want a printer to accept files of type **simple**, you must give an alternate list of content types the printer can accept. (The printer type is a good name to use if no other type is appropriate.)

Another content type name is **terminfo**. This does not refer to a particular type of file but instead refers to all the types represented in the **terminfo** database. It is not likely that any printer is capable of handling all the types listed in the database. However, this name is reserved for describing possible filter capabilities. Likewise, the content type **any** is reserved for describing the types of files a filter can accept or produce. These names should not be used as content types when adding a printer.

Specify the list of content types as follows:

/usr/lib/lpadmin -p *printername* **-I** *content-type-list*

Δ **sysadmsh** users select: Printers→Configure→Content

The *content-type-list* is a list of names separated by a comma or space. If you use spaces to separate the names, enclose the entire list (but not the **-I**) in quotes. If you do not define the types of files a printer can accept, the print service assumes it can take type **simple** and a type with the same name as the printer type (if the printer type is defined).

Character Sets or Print Wheels

Printers differ in the way they can print in different font styles. Some have changeable print wheels, some have changeable font cartridges, others have preprogrammed, selectable character sets. The print service, with your help, can minimize the impact of these differences on the users of the print service.

When adding a printer, you can specify what print wheels, font cartridges, or character sets are available with the printer. Only one of these is assumed to apply to each printer. From the point of view of the print service, however, print wheels and changeable font cartridges are the same because they require you to intervene and mount a new print wheel or font cartridge. Thus, for ease of discussion, only print wheels and character sets are mentioned.

When you list the print wheels or character sets available, you are assigning names to them. These names are for your convenience and the convenience of the users. Because different printers may have similar print wheels or character sets, you should use common names for all printers. This allows a person to submit a file for printing and to ask for a particular font style, without regard for which printer is used or whether a print wheel or selectable character set is used.

If the printer has mountable print wheels, you need only list their names. If the printer has selectable character sets, you need to list their names and map each one into a name or number that

uniquely identifies it in the **terminfo** database. You can use the following command to determine the names of the character sets listed in the **terminfo** database:

TERM=*printer-type* **tput csnm 0**

printer-type is the name of the printer type in question. The name of the 0th character set (the character set obtained by default after the printer is initialized) should be printed. Repeat the command, using 1, 2, 3, and so on in place of the 0, to see the names of the other character sets. In general, the **terminfo** names should closely match the names used in the user documentation for the printer. However, because not all manufacturers use the same names, the **terminfo** names may differ from one printer type to the next.

NOTE: For the print service to find the names in the **terminfo** database, you must specify a printer type. See the earlier section "Printer Type."

To specify a list of print wheel names when adding a printer, enter the following command:

/usr/lib/lpadmin -p *printername* **-S** *print-wheel-list*

Δ **sysadmsh** users select: Printers→Configure→Parameters

print-wheel-list is a list of names separated by a comma or space. If you use spaces to separate the names, enclose the entire list (but not the **-S**) in quotes.

To specify a list of character set names and to map them into **terminfo** names or numbers, enter the following command:

/usr/lib/lpadmin -p *printername* **-S** *character-set-list*

Δ **sysadmsh** users select: Printers→Configure→Parameters

character-set-list is also a list of names separated by a comma or space; however, each item in the list looks like one of the following:

cs*N*=*character-setname*
character-setname$_1$=*character-setname*$_2$

N in the first case is a number from 0 to 63 that identifies the number of the character set in the **terminfo** database. *character-setname*$_1$ in the second case identifies the character set by its **terminfo** name. In either case, the name to the right of the equal (=) sign is the name you choose as an alias of the character set.

NOTE: You do not have to provide a list of aliases for the character sets if the **terminfo** names are adequate. You can refer to a character set by number, by **terminfo** name, or by your alias.

For example, suppose your printer has two selectable character sets (sets #1 and #2) in addition to the standard character set (set #0). The printer type is 5310. You enter the following commands to determine the names of the selectable character sets:

```
TERM=5310 tput csnm 1
english
TERM=5310 tput csnm 2
finnish
```

The words *english* and *finnish*, are the output of the commands, the names of the selectable character sets. You feel that the name "finnish" is adequate for referring to character set #2, but better names are needed for the standard set and set #1. You enter the following command to define synonyms:

```
/usr/lib/lpadmin -p printername -S "cs0=american, english=british"
```

Δ **sysadmsh** users select: Printers→Configure→Parameters

If you do not list the print wheels or character sets that can be used with a printer, then the print service assumes the following: a printer that takes print wheels has only a single, fixed print wheel, and people cannot ask for a special print wheel when using the printer. Also, a printer that has selectable character sets can take any **cs***N* name or **terminfo** name known for the printer.

Alerting to Mount a Print Wheel

If you have printers that take changeable print wheels and you have listed the print wheels allowed on each, then users can submit a print request to use a particular print wheel. However, until it is mounted (see "Mounting a Form or Print Wheel" in this chapter), a request for a print wheel stays queued and is not printed. You could periodically monitor the number of print requests pending for a particular print wheel, but the print service provides an easier way. You can ask to be alerted when the number of requests waiting for a print wheel has exceeded some threshold.

You can choose one of several ways to receive an alert:

- You can receive an alert via electronic mail. See **mail(C)** for a description of the **mail** command.
- You can receive an alert written to whatever terminal on which you are logged in. See **write(C)** for a description of the **write** command.
- You can receive an alert through a program of your choice.
- You can receive no alerts.

NOTE: If you elect to receive no alerts, you are responsible for checking whether the proper print wheel is mounted.

In addition to the method of alerting, you can also set the number of requests that must be queued before you are alerted, and you can arrange for repeated alerts every few minutes until the print wheel is mounted. You can choose the rate of repeated alerts, or you can choose to receive only one alert per print wheel.

To arrange for alerting to the need to mount a print wheel, enter one of the following commands:

/usr/lib/lpadmin -S *print-wheelname* **-A mail -Q** *integer* **-W** *minutes*
/usr/lib/lpadmin -S *print-wheelname* **-A write -Q** *integer* **-W** *minutes*
/usr/lib/lpadmin -S *print-wheelname* **-A** *'command'* **-Q** *integer* **-W** *minutes*
/usr/lib/lpadmin -S *print-wheelname* **-A none**

Δ **sysadmsh** users select: Printers→Auxiliary→Alert

The first two commands direct the print service to send you a mail message or write the message directly to your terminal, respectively, for each alert. The third command directs the print service to run *command* for each alert. The shell environment currently in effect when you enter the third command is saved and restored for the execution of *command*; this includes the environment variables, user and group IDs, and current directory. The fourth command above directs the print service to never send you an alert when the print wheel needs to be mounted. *integer* is the number of requests that need to be waiting for the print wheel, and *minutes* is the number of minutes between repeated alerts.

NOTE: If you want mail sent or a message written to another person when a printer fault occurs, you will have to use the third command listed. Use the **-A 'mail** *user-name*' or **-A 'write** *user-name*' **option.**

Once you start receiving repeated alerts, you can direct the print service to stop sending you alerts for the current case by giving the following command:

/usr/lib/lpadmin -S *print-wheelname* **-A quiet**

Δ **sysadmsh** users select: Printers→Auxiliary→Alert

Once the print wheel is mounted and unmounted again, alerts start again if too many requests are waiting. Alerts also start again if the number of requests waiting falls below the **-Q** threshold and then rises up to the **-Q** threshold again, as when waiting requests are canceled or if the type of alerting is changed.

If *print-wheelname* is **all** in any of the commands above, the alerting condition applies to all print wheels for which an alert has already been defined.

If you do not define an alert method for a print wheel, you do not receive an alert for it. If you do define a method but do not give the **-W** option, you are alerted once for each occasion.

Fault Alerting

The print service provides a framework for detecting printer faults and alerting you. Faults can range from simple problems, such as running out of paper or ribbon or needing to replace the toner, to more serious faults, such as a local power failure or printer failure. The range of fault indicators is also broad, ranging from dropping carrier (the signal that indicates that the printer is online) to sending an XOFF, or a message. Only two classes of printer fault indicators are recognized by the print service itself: a drop in carrier and an XOFF not followed in reasonable time by an XON. You can choose one of several ways to receive an alert to a printer fault:

- You can receive an alert via electronic mail. See **mail(C)** for a description of the **mail** command.
- You can receive an alert written to the terminal on which you are logged in (any terminal). See **write(C)** for a description of the **write** command.
- You can receive an alert through a program of your choice.
- You can receive no alerts.

NOTE: If you elect to receive no alerts, you need a way of finding out about the faults and fixing them; the print service does not continue to use a printer that has a fault.

In addition to the method of alerting, you can also arrange for repeated alerts every few minutes until the fault is cleared. You can choose the rate of repeated alerts, or you can choose to receive only one alert per fault.

NOTE: Without a filter that provides better fault detection, the print service cannot automatically determine when a fault has been cleared except by trying to print another file. It assumes that a fault is cleared when it successfully prints a file. Until that time, if you have asked for only one alert per fault, you do not receive another alert. If after you have fixed a fault, but before the print service has tried printing another file, the printer faults again, or if your attempt to fix the fault did not succeed, you are not notified. Receiving repeated alerts per fault or requiring manual re-enabling of the printer (see the "Fault Recovery" section later) overcomes this problem.

To arrange for alerting to a printer fault, enter one of the following commands:

/usr/lib/lpadmin -p *printername* **-A mail -W** *minutes*
/usr/lib/lpadmin -p *printername* **-A write -W** *minutes*
/usr/lib/lpadmin -p *printername* **-A** *'command'* **-W** *minutes*
/usr/lib/lpadmin -p *printername* **-A none**

Δ **sysadmsh** users select: Printers→Configure→Errors

The first two commands direct the print service to send you a mail message or write the message directly to your terminal, respectively, for each alert. The third command directs the print service to run *command* for each alert. The shell environment currently in effect when you enter the third command is saved and restored for the execution of *command.* The environment includes environment variables, user and group IDs, and current directory. The *minutes* is the number of minutes between repeated alerts. The fourth command above directs the print service not to send you an alert when a fault occurs.

NOTE: If you want mail sent or a message written to another person when a printer fault occurs, use the third command. Use the option:

-A 'mail *username'* or **-A 'write** *username'*

Once a fault occurs and you start receiving repeated alerts, you can direct the print service to stop sending you alerts for the current fault by giving the following command:

/usr/lib/lpadmin -p *printername* **-A quiet**

Δ **sysadmsh** users select: Printers→Configure→Errors

If *printername* is **all** in any of the commands above, the alerting condition applies to all printers.

If you do not define an alert method, you receive mail once for each printer fault. If you do define a method but do not give the **-W** option, you are alerted once for each fault.

Fault Recovery

Once a printer fault is detected and you are alerted, you will probably fix the fault and get the printer ready for printing. When the printer is ready for printing again, the print service recovers in one of three ways:

- continues printing at the top of the page where printing stopped,
- restarts printing at the beginning of the print request that was active when the fault occurred, or
- waits for you to tell the print service to re-enable the printer.

NOTE: The ability to continue printing at the top of the page where printing stopped requires the use of a filter that can wait for a printer fault to be cleared before resuming properly. Such a filter probably has to have detailed knowledge of the control sequences used by the printer so it can keep track of page boundaries and know where in a file printing stopped. The default filter used by the print service cannot do this. If a proper filter is not being used, you are notified in an alert if recovery cannot proceed as you want.

To specify the way the print service should recover after a fault has been cleared, enter one of the following commands:

/usr/lib/lpadmin -p *printername* **-F continue**
/usr/lib/lpadmin -p *printername* **-F beginning**
/usr/lib/lpadmin -p *printername* **-F wait**

Δ **sysadmsh** users select: Printers→Configure→Errors

These direct the print service, respectively, to continue at the top of the page, restart from the beginning, or wait for you to enter an **enable** command to re-enable the printer (see the "Enabling and Disabling Printer" section earlier in this chapter for information on the **enable** command).

If you do not specify how the print service is to resume after a printer fault, it tries to continue at the top of the page where printing stopped, or failing that, at the beginning of the print request.

If the recovery is **continue** but the interface program does not stay running so that it can detect when the printer fault was cleared, printing is attempted every few minutes until it succeeds. You can force the LP print service to retry immediately by issuing an **enable** command.

Default Printing Attributes

When a user submits a request to print a file, the page size, character pitch, and line pitch (that is, print spacing) are normally determined from the form that is printed on. If the user does not require a form, he or she can give the page size and print spacing to use. However, if he or she gives neither a form to use nor the page size and print spacing, defaults are used.

You can set the defaults for each printer. This can also serve to make submitting a print request easier, by designating different printers as having different default page sizes or print spacing. Users then simply route their file to the appropriate printer to get the style output they want. For example, you can have one printer dedicated to printing wide (132 column) output, another printing normal (80 column by 66 lines) output, yet another printing letter quality (12 characters per inch, 8 lines per inch).

You can independently specify four default settings: page width, page length, character pitch, and line pitch. You can scale these to fit your needs. The first two can be given in columns and lines, inches, or centimeters. The last two can be given as characters and lines per inch or per centimeter. In addition, the character pitch can be specified as **pica** for 10 characters per inch (cpi), **elite** for 12 cpi, or **compressed** for the maximum cpi the printer can provide (up to a limit of 30 cpi).

Set the defaults using one or more of the following commands:

/usr/lib/lpadmin -p *printername* **-o width=***scaled-number*
/usr/lib/lpadmin -p *printername* **-o length=***scaled-number*
/usr/lib/lpadmin -p *printername* **-o cpi=***scaled-number*
/usr/lib/lpadmin -p *printername* **-o lpi=***scaled-number*

Δ **sysadmsh** users select: Printers→Configure→Parameters

Add the letter "i" to *scaled-number* to indicate inches, or the letter "c" to indicate centimeters. The letter "i" for character pitch (cpi) or line pitch (lpi) is redundant. You can also give **pica**, **elite**, or **compressed** instead of a number for the character pitch.

If you do not provide defaults, the page size and print spacing are those available when the printer is initialized. You can find out what the defaults are by first defining the printer configuration without providing your own defaults, then using the **lpstat** program to display the printer configuration. The command

lpstat -p *printername* **-l**

reports the default page size and print spacing. If you have not provided the defaults, the reported defaults are calculated from the **terminfo** database entry for the printer. Obviously, this requires you to have provided a printer type in the printer configuration.

Setting Up RTS/CTS Protocol Serial Printers

The RTS and CTS lines for the RS-232 serial interface were originally intended as handshaking signals between a Data Terminal Equipment (DTE) device (computer, printer, etc,) and a Data Communications Equipment (DCE) device (almost always a modem). The RTS (Ready To Send) line is asserted by the DTE when it is ready to send data to the DCE. The DCE asserts the CTS (Clear To Send) line when it was ready to receive data. If the CTS line goes low, then the DTE should stop sending data until CTS goes high again.

The operating system also uses the RTS line for handshaking in the other direction. If the printer sees that its input buffer is nearly full, it will lower the CTS line. The serial driver will then stop sending, and wait for the printer to catch up. The operating system will raise the CTS line when it is ready for more data.

Many printers use the DTR line for handshaking rather than RTS or CTS. For these devices, the cable must be wired to connect the printer's DTR pin to the computer's CTS pin (see Figure 12-3).

To set up for RTS/CTS flow control, do the following:

1. Use the modem-control port (e.g. */dev/tty1A*). If you plan to use the spooler to access this printer, make sure you specify the modem control port rather than one of the standard serial devices displayed when you use the **sysadmsh Printers→Configure→Parameters** selection asks you to enter a device name.

2. Make sure **stty** settings include **-ixon -ixoff -clocal rtsflow ctsflow**.

3. For a device that uses the RTS and/or CTS lines for handshaking, the cable should be wired as shown in Figure 12-2.

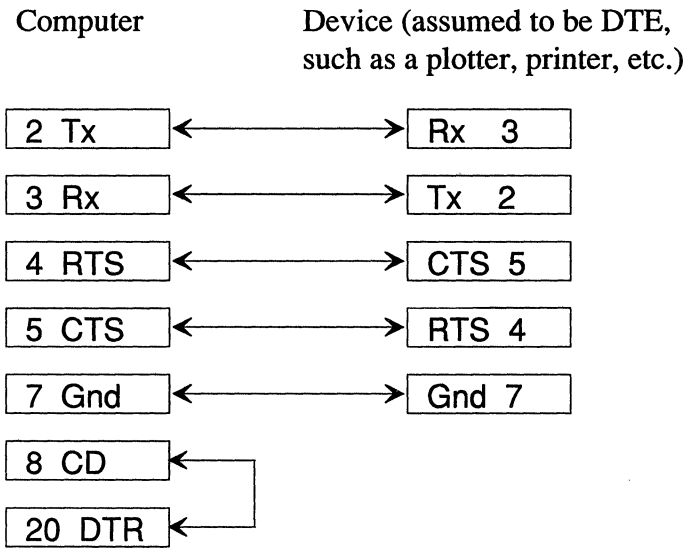


Figure 12-2. RTS/CTS Handshaking

4. If the device uses the DTR line for handshaking, the cabling should be as shown in Figure 12-3.

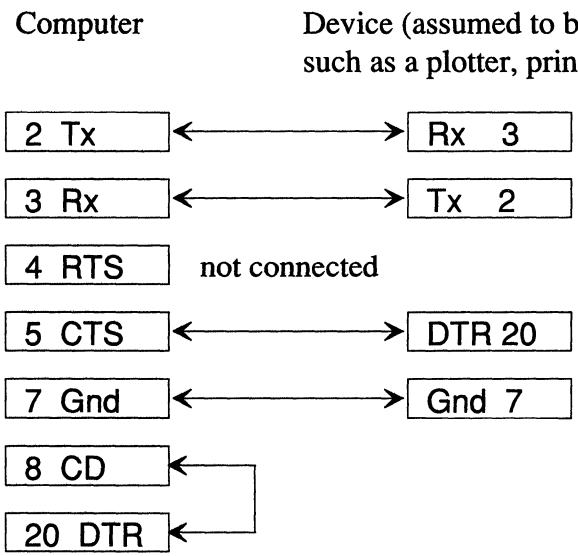


Figure 12-3. DTR Handshaking

5. If the information contained here does not solve the problem, try removing **rtsflow** from the **stty** command string.

Using a Printer without the Spooler

If you use a printer without the spooler, any **stty** settings you have specified for use with that printer do not stay in effect. The spooler opens the file and then runs the **stty** commands as specified in the printer interface script. To use a printer without the spooler, follow the instructions in this section.

While logged in as root, give the following commands or insert them into the initialization file */etc/rc2.d/S80lp* before the line that calls **/usr/lib/lpsched**:

(stty *baud* **ixon ixoff-ixany ; cat > /dev/null) < /dev/tty***n* **&**

where *baud* is the baud rate of the printer, and tty*n* is the serial device name. This command sets the stty options and holds the port open for use without the spooler.

NOTE: If you ever need to enable the port, make sure you kill this process first. This command does not work from a C-shell (**csh**). It returns the message:

```
stty: invalid option.  csh
```

In addition, with certain multiport cards, it is necessary to add a **sleep** command after the initialization program supplied with the card, *initprogram*, followed by the **stty** holdopen command:

initprogram **&**
sleep 3

The above script is specific to serial printers. A more general one that works for both parallel and serial is:

(stty *baud* **onlcr; while : ; do sleep 3600; done) < /dev/lp1 &**

where the stty settings you desire follow the word stty (that is, baud rate, ixon, ixoff, -ixany, onclr, and so on) and lp1 is replaced by the device name of the printer (such as, *tty1a* or *lp2*)

Chapter 13

Using Floppy Disks and Tape Drives

An important part of any computer system is the ability to offload files and restore them when needed. There are several types of media used to store and recall files. Among these are floppy disks and magnetic tape devices. This chapter explains how to install and use the above types of storage media with your system. Your system should come with at least a floppy disk drive already installed and ready to run. This chapter provides instructions on how to add tape drives and how to use floppy disks.

Using Cartridge Tape Drives

A tape cartridge drive is a mass storage device that uses 1/4 inch tape cartridges to store data. It is also referred to as a QIC (quarter inch cartridge) tape drive. A tape cartridge can hold many times the data that can be stored on floppies, making it much more useful for large backup operations.

The drives that are supported are listed in the *Release Notes*. For hardware-specific information, refer to the manual for your drive and **tape**(HW) in this guide.

Installation and Configuration

Read your tape drive hardware manual for physical installation instructions and general information.

To add a tape drive, log in as root and enter the following:

mkdev tape

Δ **sysadmsh** users select: System→Hardware→Tape

The following menu is displayed:

```
**** Tape Drive Configuration Program ****

        1. Install a Tape Drive
        2. Remove a Tape Drive

Enter an option or 'q' to quit:
```

Enter '1' to add the drive. You are then asked to select the type of tape drive you have installed:

```
        1. Install Cartridge Tape Driver
        2. Install Mini-Cartridge Tape Driver
        3. Install QIC-40 Tape Driver
        4. Install SCSI Tape Driver

Enter an option or 'q' to quit:
```

The subsections that follow describe the configuration requirements for each drive type. Be sure and consult the sections on "Kernel Relinking" and "Boot Messages" after following the instructions for your drive type.

Cartridge Tape

The "Cartridge Tape Driver" selection refers to the QIC-02-type full-size cartridge tape drives. You need to know the following technical information before you install your QIC tape drive:

- The interrupt number. The default interrupt is set on your tape drive controller card. If this number conflicts with one that is already in use, you must change the setting on the card. Interrupts 0, 1, and 6 are always used by the operating system, even if no other devices are present. If you set the interrupt to anything other than the default, write down the setting you chose, as you need to specify it when you run the **mkdev** utility.

- The DMA Channel and base I/O address. There are also default settings on your tape drive controller card for DMA channel and base I/O address. If you need to change these because of a conflict with existing hardware, note the settings you select and specify them when you run **mkdev**(ADM).

If you are using the default settings on your controller card, you can **q** at the second menu to use the default tape parameters.

If you modified the default settings on your controller card, select the option to "Modify Current Tape Parameters" at the second menu. Next, you see a menu with the default tape parameters. Change any tape parameters here that you changed on your controller board. Note that if you changed the base address, you must enter an "0x" before the number if it is a hexadecimal address. If you do not specify the "0x", the system assumes that the address is decimal. Also, note that if you choose interrupt 2 on your controller, you must specify interrupt 25 when you modify your tape parameters. The software interrupt 25 corresponds to the hardware interrupt 2. All other interrupts use the same number in software as in the hardware.

Consult the *Release Notes* for information about interrupts, DMA channels and addresses.

Mini-Cartridge

Mini tape drives use the floppy disk drive controller and are significantly different from standard QIC tape drives. These units are the so-called "floppy" or "Irwin" tape units. They are not configurable and do not require parameters to be entered. In addition, mini tapes must be formatted before they can be used. There are also some differences in the installation of mini tapes.

First make sure that your drive is correctly jumpered. The correct setting may be different for different brands of machines. See your hardware documentation and your *Release Notes* for more information.

QIC-40

These units are specialized mini-cartridge units that use a different format from the mini-cartridge units described above. They are also non-configurable.

SCSI

SCSI tape drives are attached to a SCSI adapter. You are asked to provide the device ID number (0-7), the number of the adapter the drive is attached to, and the logical unit number of the device (LUN).

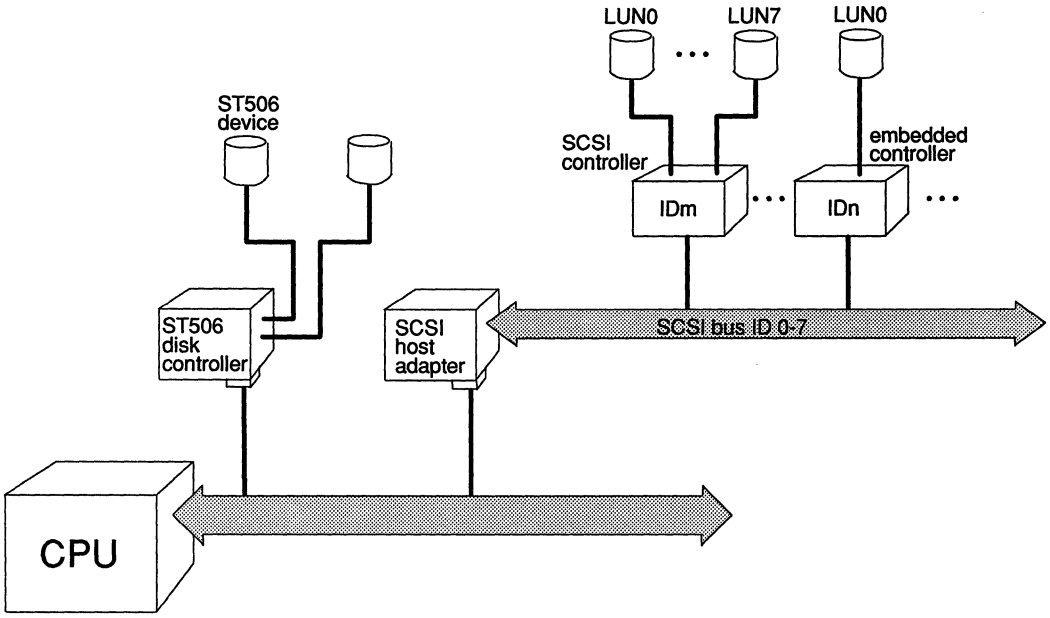


Figure 13-1. SCSI Configuration Schematic

As shown in Figure 13-1, a SCSI host adapter (HA) translates signals from the CPU bus to the SCSI bus. A SCSI controller is known as a SCSI ID. A SCSI device is referenced by a logical unit number (LUN).

To configure a SCSI tape device, you must know:

- the ID number of the controller (0-7) on the host adapter; the host adapter itself is usually ID 7, giving it the highest priority on the SCSI bus
- the host adapter number (0 or 1)

Because the tape drive and its controller are one unit (referred to as "embedded") the LUN or logical unit number is simply "0." The **mkdev tape** prompts for these values appear as follows:

```
What is the ID of the controller for this device?
Select 0-7, or 'h' for help, or 'q' to quit:

Which SCSI host adapter supports this device?
Select 0 or 1, or 'h' for help, or 'q' to quit:

What is the LUN of this device?
Select 0-7, or 'h' for help, or 'q' to quit:
```


Kernel Relinking

After you select the driver to be installed and provide any additional information, you are prompted to permit relinking of the kernel. The tape drive unit will be available for use after rebooting.

Boot Messages

When the kernel recognizes a tape drive (and when the driver is linked into the kernel) a message is always displayed at boot time indicating the device is present. This information can also be displayed using **hwconfig**(C). Table 13.1 contains the messages displayed indicating the drive type.

Table 13.1.

Tape Drive Boot Messages

Type	Boot Display Message				
QIC-02	`%tape`	-	-	-	`type=W`
mini	`%ctmini`	-	-	-	`type=ir`
QIC-40	`%ctmini`	-	-	-	`type=qic40`
SCSI	`%tape`	-	-	-	`type=S ha=0 id=2 lun=0`

Editing /etc/default/tar

After you install your tape drive, you must enter the correct size setting in the */etc/default/tar* file. When you edit the file, you see several entries for various default devices. Figure 13-1 shows the */etc/default/tar* file provided with your distribution.

```
# device                    block   size    tape
archive0=/dev/rfd048ds9     18      360     n
archive1=/dev/rfd148ds9     18      360     n
archive2=/dev/rfd096ds15    10      1200    n
archive3=/dev/rfd196ds15    10      1200    n
archive4=/dev/rfd096ds9     18      720     n
archive5=/dev/rfd196ds9     18      720     n
archive6=/dev/rfd0135ds18   18      1440    n
archive7=/dev/rfd1135ds18   18      1440    n
archive8=/dev/rct0          20      0       y
archive9=/dev/rctmini       20      0       y
# The default device...
archive=/dev/rfd096ds15     10      1200    n
```

Figure 13-2. /etc/default/tar File

QIC Cartridge Drives

The */dev/rct0* entry is used to access the QIC cartridge tape drive. The cartridge sizes are indicated in Table 13.2.

Table 13.2.

QIC Cartridge Sizes

Feet	Entry in **Size** Field
300	30000
450	45000
600	60000

Mini-Cartridge Drives

Find the entry in your */etc/default/tar* file for */dev/rctmini*. In the above sample file, this is **archive9**. Note that the size value for **rctmini** is 0. If you plan to use the default file, you must change this entry when you install your rctmini device. The correct number for your rctmini device varies with the size of the tape you use.

Table 13.3.

Mini-cartridge Sizes

Tape Size	Actual Capacity	Entry in **Size** field
10 megabyte	8 MB	8000
20 megabyte	17 MB	17000
40 megabyte	35 MB	35000
80 megabyte	72 MB	72000

The utilities **xbackup** and **xrestore** have similar files and entries. For more information on default files, see **default**(F) and the manual entry for the particular backup or restore command.

Archiving Files on Tape

You use a tape drive much like a floppy, but the volume of data stored is much greater. Tapes are much better for storing (backing up) entire filesystems. The **tar**(C) command is the recommended archive program for users and is best used for general archiving/transporting of files. Other programs such as **backup**(ADM) and **restore**(ADM) are meant for system administrators making copies of entire filesystems. Consult "Backing Up Filesystems" in this guide for making regular backups of filesystems.

The **cpio**(C) command is a general purpose archive program that uses a different format than **tar**. The **dd**(C) program is used to transfer or convert archives of unusual format; the input and output format can be specified on the command line.

The tar Command

The **tar** command is useful for making a backup copy of entire directories. The command has the syntax:

tar cvf *devicefile files*

The *devicefile* is the file name that corresponds to the cartridge drive. *files* are the names of the files or directories to be copied. For example, to copy all the files in the directory */u/bogart* to the cartridge drive **/dev/rct0**, enter:

tar cvf /dev/rct0 /u/bogart

Δ **sysadmsh** users select: Media→Archive

To restore files stored on tape, insert the cartridge containing the files or directories you wish to restore and enter the following command:

tar xvf *devicefile*

Δ **sysadmsh** users select: Media→Extract

tar restores all the files on the tape to the original directory.

Tape Drive Maintenance

The **tape**(C) utility performs various tape maintenance operations on all tape drives. **tape** sends commands and receives status from the tape drive. The basic form of the command is:

tape *command* [devicefile]

For example, to rewind a cartridge tape device, enter:

tape rewind

Other commands are:

erase Erase tape cartridge. Also re-tensions.

reset Reset tape controller and tape drive. Clears error conditions and returns tape subsystem to power up state.

reten Re-tension tape cartridge. Should be used periodically to remedy slack tape problems, generally resulting in an unusually large number of tape errors.

After certain tape operations are executed, the system returns a prompt before the tape controller has finished its operation. If you enter another tape command too quickly, the message "device busy" is displayed until the tape device is finished with its previous operation.

You should clean the tape drive heads and re-tension cartridges to keep it operating error-free.

Tape Formatting

Tape cartridges used with the mini tape drive (ctmini) must be formatted before use. With the exception of QIC-40 and QIC-80 tapes, they must also be servo-written before formatting. Use the **tape**(C) utility to format and servo-write a cartridge tape. If a tape has been previously servo-written, it must be erased with a bulk-eraser before being servo-written again. The following command will servo-write a blank, bulk-erased tape:

```
tape servo
```

The following command formats a ctmini tape cartridge:

```
tape format
```

See also **tape**(HW) and **tape**(C) for more information.

NOTE: Do not specify the raw device (example */dev/rft0*) when using the **tape** command.

Tape Driver Error Correction Code (ECC) Support

Tape ECC is also supported. The ECC tape device node, */dev/erct0* is automatically created when you run **mkdev tape**. In order to use ECC, you must read and write from this device, not the normal */dev/rct0*. Users with tape drives that support cartridges larger than 60 MB should consider editing the */etc/default/tar* file and substituting */dev/erct0* for their normal tape device.

The error recovery scheme is 2/64, which means that two 512-byte blocks out of every 64 blocks can go bad and the driver will correct them. The probability of error with ECC is $1:10^{14}$. Standard drives have a error probability of $1:10^{9}$.

Be sure and label tapes that are created with the ECC device; these tapes cannot be read by standard devices. In addition, if transporting data from one machine to another, it is advisable to use the ECC device only if the target machine supports the ECC scheme.

Using Floppy Disks

Floppy disks are the most convenient form of storage media. Depending on your floppy disk drive, you may be able to store from 360 kilobytes to 1.4 megabytes on a single disk. Floppy disks can be used for simple data storage in **tar**, **cpio**, **dd** or **dump** formats or you can make a mountable filesystem on a floppy disk. The following sections explain how to use floppys for data storage and as extra filesystem space.

Formatting Floppy Disks

Floppy disks must be formatted before they can be used. The UNIX command to format a floppy disk is:

format /dev/*floppy-device*

Δ **sysadmsh** users select: Media→Format

The floppy device you specify in the command relates to the type of disk drive and floppy you are using. For example, if you have a high density 5.25 inch floppy disk drive, you can use it in high density mode (96 tracks per inch) or in low density mode (48 tpi). If you have high density floppies to use with your drive, the floppy device to specify is:

/dev/rfd096

In the above example, *rfd* indicates the raw floppy device, *0* indicates that this is the primary floppy drive, and *96* indicates high density mode. Similarly, if you wish to use low density floppies and the low density mode of the floppy drive, the device name is:

/dev/rfd048

In the above example, *48* indicates the low density mode of floppy drive *0*.

/etc/default/format file

You can also define a default format device by adding an entry to the file */etc/default/format*. For example:

```
DEVICE=/dev/rfd096ds15
```

After adding the above line, you no longer have to specify the device name. In addition, it is possible to define that all floppies be verified, which confirms that the data on the floppy is readable. (This can also be specified on the command line with the **-y** option.) Automatic verification can be specified by the following entry:

```
VERIFY=Y
```

If this entry is placed in */etc/default/format*, all floppies formatted with the **format** command are verified. (To override verification, use the **-n** on the command line.)

Refer to **format**(C) for more details.

Creating an Emergency Boot Floppy Set

mkdev(ADM) provides a utility to create an Emergency Boot Floppy Set to allow you to restore a corrupted root filesystem without reinstalling the operating system. If you have more than one system, you should make one Emergency Boot Floppy Set for each machine. Because each machine has a unique "Emergency" set, a set made on one system will not work with any other system. Be sure to keep these diskettes separate; if you use an emergency floppy set on the wrong machine, it will not work and further corruption may result.

To create the floppy set, **mkdev fd** uses a menu-driven program to select the disk format and filesystem type. The program actually generates three types of disks: simple filesystem, and the two used in the Emergency Boot Floppy Set, bootable only and root filesystem only. You must create one bootable and root filesystem disk to make up your set. The formats supported are: 48 tpi, 96 tpi-15 sectors/track in the 5 1/4 inch format, and 135 tpi-9 sectors/track in the 3 1/2 inch format. To create the floppies, follow this procedure:

1. Log in as **root** and enter:

   mkdev fd

   Δ **sysadmsh** users select: Filesystems→Floppy

2. You see the following display:

```
Floppy Disk Filesystem Creation Program

        Choices for type of floppy filesystem.

        1. 48tpi, double sided, 9 sectors per track
        2. 96tpi, double sided, 15 sectors per track
        3. 135tpi, double sided, 9 sectors per track
        4. 135tpi, double sided, 18 sectors per track

        Enter an option or enter q to quit:
```

Enter the number of the disk type desired and press **Return**.

3. Next you see:

```
Choices for contents of floppy filesystem.

1. Filesystem
2. Bootable only          (96ds15 and 135ds18 only)
3. Root filesystem only   (96ds15 and 135ds18 only)

Enter an option or enter q to quit:
```

Create the bootable disk first; enter **2** and press **Return**.

4. You see the following prompt:

```
Insert a type floppy into drive 0.
Press Return to continue or enter q to quit:
```

Press **Return**.

5. The following prompt is displayed:

```
Would you like to format the floppy first? (y/n)
```

If you have already formatted the floppy, enter **n** and the filesystem is immediately created. If the floppy has not yet been formatted, enter **y** and you see:

```
formatting /dev/type
track 00 head 0
```

The track and head numbers will count up as the floppy is formatted. (If */etc/default/format* contains "VERIFY=Y", the format will also be verified after formatting.)

6. The following is displayed:

```
Successfully created filesystem.

Copying files to /dev/type ...
```

The bootable disk is then generated by creating a filesystem and copying the relevant files from the root filesystem. **mkdev** also checks the filesystem with **fsck**(ADM); messages similar to the filesystem check displayed at boot time are displayed.

7. The following message is displayed when the disk is ready:

```
type floppy created and checked successfully
```

8. You are then returned to the main menu. You should now create the root filesystem diskette. Enter **3** and press **Return**.

9. You see the following prompt:

```
Insert a type floppy into drive 0.
Press Return to continue or enter q to quit:
```

Press **Return**.

10. The following prompt is displayed:

```
Would you like to format the floppy first? (y/n)
```

    If you have already formatted the floppy, enter **n** and the filesystem is immediately created. If the floppy has not yet been formatted, enter **y** and you see the formatting messages described earlier.

11. The following messages are displayed:

```
Copying files to /dev/type root filesystem ...

Copying special files to /dev/type root filesystem ...
```

12. As with the bootable floppy, **mkdev** also checks the filesystem with **fsck**(ADM); similar messages are displayed.

13. The following message is displayed when the disk is ready:

```
type floppy created and checked successfully
```

Store these diskettes in a safe place. You will need them if your system becomes corrupted and is no longer bootable.

Chapter 14
Using Bus Cards

The bus (or "motherboard") of your computer is the center of your system. Every system administrator must deal with the bus and the hardware associated with it. To find the bus on your system, you must generally remove the shell from the main body of your computer. Generally, you find a large circuit board with expansion slots for extra boards. These boards are commonly known as *Bus Cards*.

Bus cards can be extra memory for your system, internal modems, multi-port serial boards for extra terminals, controller boards for peripheral devices such as hard disks tape drives, control cards for monitors with color and graphics capabilities, mouse controllers, or other devices. In this chapter we explain a little about bus cards and how to install them with your UNIX system. Installation of most devices with bus cards is explained in detail in other chapters of this Guide, so we will deal only with the basics of bus cards in this chapter.

Installing Bus Cards

To install a bus card, you must first shut down the operating system and power down the system. Make sure that the computer is unplugged or you may injure yourself. Before you begin working on the computer, ground yourself by touching a metal object close at hand that is not the computer. Static electricity that builds up and jumps from your hand when you touch the hardware inside the computer can ruin your equipment.

Dip Switches and Jumpers

Before you plug your board into the bus, make sure that there are no settings on the board that must be changed. Again, your hardware documentation that comes with the board should list the default settings and how to change them. Generally, to change the settings of a board, there are dip switches and "jumpers." Dip switches operate in "down" and "up" positions. Your hardware documentation should list the correct settings if your board has these switches. Jumpers are clips that slide over metal posts that stick out of the board to make a connection. You can change the settings on a board by moving the jumper to connect a different pair of posts. Again, your hardware documentation should provide you with specific instructions for jumper settings on your hardware.

NOTE: Note that the operating system is designed to work with most hardware using default settings. You will rarely have to change the settings on a board.

Installing the Hardware

Carefully perform any steps necessary to expose the expansion slots on your computer. Your hardware documentation should explain this in detail. Once you can examine this area, note the number of available spaces for bus cards. A new system will have up to 8 or 10 available slots. Note that some slots are longer than others. There are both short and long cards. Short cards are about half as long as long cards. Generally there are 2 to 3 short slots and the rest are long slots. Find a slot that fits your board and gently but firmly plug the board into the slot in the bus. The board should have a tab on one side that fits into the slot on the bus. Bus cards only fit one way.

Some bus cards have a port that should face the outside of the computer. As stated before, bus cards only fit into the system one way. There may be a small plate covering an opening in the computer held on with a small screw. You can remove this cover plate if you need to. Boards such as modems, serial and parallel cards, and external device control cards will require this.

When you are done, replace the shell for your computer, and turn it on and boot. You may first need to use the manufacturer's setup program as described below to change the system's configuration before you can use the new hardware.

Using the Manufacturer's Setup Disk

Your computer should come with a manufacturer's setup program on a bootable floppy disk. Copy this disk for use and keep the original in a safe place. This disk is used to configure the permanent memory on your computer to describe the system hardware setup. Whenever you add a major device, like an extra hard disk or an extra serial card, you may need to run your setup program to tell your computer about the new hardware. Some computers automatically recognize the presence of new hardware. Your manufacturer's documentation should let you know if you need to run this software.

Adding Additional Memory

You can improve system performance and run larger programs by increasing the amount of internal memory.

To increase internal memory follow these steps:

1. Turn off your computer.
2. Install extended memory according to the manufacturer's instructions. Make sure you have set all switches as noted in the instructions.
3. Boot the operating system. The boot screen details how the additional memory has affected your system.
4. Some features may have been expanded. For example, you may have:
   - More multiscreens
   - More buffers
   - A larger maximum user process size

The number of multiscreens may be unchanged. Since the number of multiscreens can be set by the user, you may have already set a specific limit to the number of multiscreens available. If you have not set a limit to the number of multiscreens then you are already using the maximum number of multiscreens that the system allows.

The number of buffers may also be unchanged. Since the number of buffers can also be set by the user, you may have already set a specific limit to the number of buffers available. If you have not set a limit to the number of buffers then you are already using the maximum number of buffers that the Operating System allows.

If the maximum user process size is unchanged, then it is now limited by the size of the *swap* filesystem instead of the amount of internal memory. You can:

- Reinstall the operating system and increase the size of the *swap* space.
- Change the process so that it runs without being swapped. Refer to **proctl** (S) for details.

You can follow the same procedure if you wish to remove internal memory from the system.

If the memory hardware reports an error, the following message is displayed:

```
PANIC: memory parity error
```

You then see the software reboot message:

```
     ** Safe to Power Off **
                - or -
** Press Any Key to Reboot **
```

If the system repeatedly panics from parity errors, consider replacing the memory chips.

NOTE: Some machines have a hardware limitation on the maximum amount of memory that can be installed. Refer to your computer hardware manual to determine the maximum amount of memory you can install.

The Operating System uses only "extended", not "expanded" memory.

Chapter 15
Using a Mouse

This chapter deals with the basics of installing any brand or type of mouse interface with your system. Using a mouse can be a great convenience for users and developers alike. For this reason, support is provided for both serial and bus mouse hardware.

Installing the Hardware

Consult your hardware manufacturer's documentation for specific instructions on hardware configuration. Note the brand and type of your mouse and whether it is attached to a serial port or directly to the system bus. For more information about the system bus, see the chapter in this Guide called "Using Bus Cards." You will need to know this information when you configure your software to accept the mouse.

NOTE: Please note the following restrictions regarding mouse usage:

- The Microsoft Bus mouse cannot be configured using interrupt vector 2; use 3, 4 or 5 instead.
- Do not use the **usemouse** utility while in single-user (maintenance) mode.
- You cannot invoke the System→Terminate (**shutdown**) using the **usemouse** utility.

Installing a Mouse

To install a mouse on your system, you must perform the following steps:

1. Install the mouse according to the manufacturer's instructions.
2. Make sure your link kit is installed and functioning correctly. The mouse drivers cannot be installed without the Link Kit. (The link kit is installed using **custom**(ADM).)
3. Log in as root and input the following command:

   mkdev mouse

   Δ **sysadmsh** users select: System→Configure→Hardware→Mouse

   And you see the Mouse Initialization Menu:

```
        Mouse Initialization Program

        1. Display current configuration
        2. Add a mouse to the system
        3. Remove a mouse from the system
        4. Associate a terminal with an existing mouse
        5. Disassociate a terminal from an existing mouse
        6. Remove the mouse drivers from the kernel

Select an option or enter q to quit:
```

   To install a mouse, select option **2** and press **Return**. The other options allow you to change your mouse configuration at any time. For example, you can add or remove additional mice on your system or change the terminals that are allowed to receive input from an existing mouse.

4. Next, you must specify the type of mouse you will use. You see the menu:

```
The following mouse devices are supported:

1. Logitech Serial Mouse
2. Microsoft Serial Mouse
3. Mouse Systems PC Mouse
4. Microsoft Bus mouse
5. Olivetti Bus Mouse
6. Logitech Bus Mouse
7. Keyboard Mouse

Select an option or press 'q' to return to the previous menu:
```

Enter the number corresponding to the mouse you wish to install and press **Return**.

5. Next, you see:

```
mouse_type is currently configured
to attach to the system on /dev/tty
Do you want to install this mouse on a different port? (y/n)
```

Enter **y** if you wish to change the default.

6. If you are installing a busmouse, you are asked to select the configuration for the busmouse card. If you are installing a serial mouse, skip this step and go directly to Step 7. If you chose a busmouse, you see the message:

```
Busmouse Configuration

1. Display current busmouse parameters
2. Modify current busmouse parameters
3. Select previous busmouse parameters
4. Select default busmouse parameters

Enter an option or q to quit:
```

If you wish to use the default busmouse parameters, select **4**. The current parameters are displayed and you can press **q** to quit this menu. The default busmouse selection auto-configures your busmouse. If you change the interrupt vector, note that using interrupt vector 5 will conflict with a cartridge tape device (using the same vector) if both devices are in use at the same time. (This is also true of the */dev/lp2* parallel device.)

7. If you have previously installed a mouse of any sort on your system, the driver for the mouse devices should already be linked with your kernel and you should proceed to Step 11. If you have never installed a mouse on your system, or the driver is not present in the kernel, you see the following messages. Note that these messages may take a few minutes to appear on your screen:

```
Updating system configuration...
You must create a new kernel to effect the driver change you specified.
Do you wish to create a new kernel now? (y/n/q)
```

   Answer **y** to add the mouse device driver to your kernel.

8. Next, you see:

```
      The UNIX operating system will now be rebuilt.
      This will take a few minutes.  Please wait.

      Root for this system build is /.
```

   As part of the linking process, you see the following messages:

```
      The UNIX kernel has been rebuilt.
Do you want this kernel to boot by default? (y/n)
```

   Answer **y** if you want this kernel to be used every time you boot the system.

9. The following is displayed:

```
Backing up /unix to /unix.old.
Installing new /unix.

The kernel environment includes device node files and /etc/inittab.
The new kernel may require changes to /etc/inittab or device nodes.

Do you want the kernel environment rebuilt? (y/n)
```

   Enter **y**.

10. The following is displayed:

```
The new kernel has been successfully linked and installed.
To activate it, reboot your system.

Setting up new kernel environment.
```

You have now installed the mouse drivers in your kernel.

11. Next, you are asked to specify the terminals and multiscreens that will be allowed to accept input from the mouse. Do not attempt to allow mouse input on any tty where any mice are physically connected or you will receive an error message. You may choose to allow any or all other terminals and console multiscreens to use the mouse. Entering the word "multiscreen" will associate all of the console multiscreens.

Note that only one mouse can be allowed for input on a given tty.

For more information on sharing the mouse between several terminals or multiscreens, see "Using the Mouse." You see:

```
This mouse may be configured for use on any of the system's
terminals and multiscreens.  The multiscreens and terminals
that will be associated with this mouse need to be specified.

Specify them by entering, at the following prompt, all the
ttys to be associated with this mouse.  Entering the word
"multiscreen" will associate all of the console multiscreens.

Enter a list of terminals (e.g. tty1a tty2a multiscreen)
or enter q to quit.  Press return when finished:
```

Press **Return** when you have entered all the devices desired.

```
Do you want to use the <mouse_type> on any other terminals?
(y/n)
```

Note that in above example, *mouse_type* will be replaced with the brand or type of mouse you specified earlier in the procedure. Respond **n** if no other terminals will be allowed to receive mouse input. If you answer **y** you are returned to the screen prompting for a list of terminals.

12. Finally, you are returned to the main mouse menu again. If you have no changes to make to your mouse configuration at this time, enter **q** to quit and press **Return**.

    Note that you can invoke **mkdev mouse** at any time to allow or prevent input on different terminals, remove mice or check your current configuration.

Removing a Mouse

Removal of any mouse or the mouse drivers on your system is an exact reversal of the process of installing a mouse. Choose the menu options to remove rather than to add a mouse.

Using the Mouse

Use of a mouse is automatic. If a program or utility accepts mouse input and the terminal is allowed to use the mouse, you simply invoke the program and the mouse works. If the terminal or multiscreen is not allowed to use the mouse, or the program is not configured to accept mouse input, using the mouse has no effect.

Using the Mouse with Multiscreens

Multiscreens (on monitors attached to video cards in the bus) provide the most convenient method for using the mouse. If a mouse is associated with the multiscreens on your main system console, (typically, a monitor attached to a video card in the system bus) the mouse input is associated with the current active multiscreen. For example, if your system has four multiscreens enabled on the main system console and all those screens are allowed to use the mouse, the input from the mouse goes to the program running on the active multiscreen.

Remember that programs that do not accept mouse input will be unaffected by moving the mouse, even on a mouse-allowed multiscreen.

Serial (terminal) multiscreens and serial consoles can also be configured to use the mouse.

Using a Mouse with Keyboard-based Programs

The **usemouse**(C) utility is used to map mouse movements and operations to keystrokes used by keyboard-based programs. Refer to the **usemouse**(C) manual page for complete information.

Chapter 16
Setting Up Electronic Mail

The operating system uses MMDF (the Multi-channel Memorandum Distribution Facility, version IIb, update #32) to route mail locally and over Micnet, UUCP, or other networks that provide MMDF support.

MMDF is automatically configured for local (one system) mail when the system is installed. If you did not install the entire distribution, MMDF is part of the MAIL package, which you can install using the **custom**(ADM) utility.

MMDF is a very versatile and configurable mail routing system. The rest of this chapter explains:

- how to tailor your MMDF system to your environment
- how to rebuild the MMDF hashed database whenever you change alias or routing information
- how to maintain your MMDF system and how to deal with problems

The section "How MMDF Works" explains the concepts crucial to an understanding of how MMDF functions. The later sections assume that you have familiarized yourself with these concepts.

How MMDF Works

In order to understand how to configure MMDF and how it functions, there are four basic components you need to be familiar with: domains, channels, aliases, and the MMDF configuration file (*mmdftailor*).

Domains

A domain is a logical name for a group of machines, each known as a host. For example, on the DARPA INTERNET (A largely TCP/IP-based network), the following domains are defined:

COM	commercial institutions
EDU	educational and research institutions
GOV	government institutions
MIL	Military institutions

The following are some examples:

sco.COM	The Santa Cruz Operation
ucscc.ucsc.EDU	a computer at UC Santa Cruz
seismo.css.GOV	a computer at the Center for Seismographic Studies

The above names are "fully qualified" names, meaning that they contain the name of the machine and the domain in order of decreasing specifics:

machine.department.company.domain

For example, a host such as *mother.comp.nostromo.COM* reads: machine "mother" at the computer center at site Nostromo in the COM domain.

Example: Site ConAm

If the site with MMDF is a machine with no connections to any other machines, no "tailoring" is necessary for normal mail traffic between users on the system. However, a system that is able to send mail offsite or to other machines at the same site must decide how to divide up the hosts into domains. Suppose a company called ConAm has three machines. They are:

conam	the main machine that has connections to the outside world
guardian	a machine connected to conam via Micnet
colossus	another machine connected to conam via Micnet

conam also has two UUCP connections:

obie	a machine connected to conam via UUCP
proteus	another UUCP connection to conam

proteus also has a connection to the DARPA INTERNET.

Figure 16-1 is a schematic of the network configuration: the box indicates the onsite machines of ConAm.

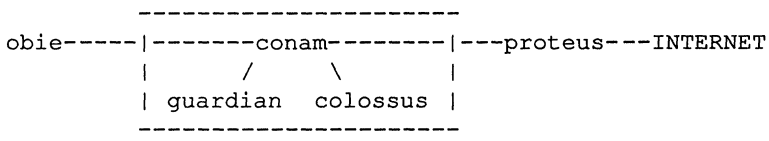


Figure 16-1. Network Configuration for Site ConAM

We can now identify the following domains:

local:	conam machine by itself
Micnet:	guardian, colossus
UUCP:	obie, proteus
INTERNET:	via proteus

The "fully-qualified" names for these hosts are:

conam.conam.UUCP
guardian.conam.UUCP
colossus.conam.UUCP
obie.UUCP
proteus.COM

The conam company machines follow the convention of naming things as *machine.company.domain*, while the other machines, being the only machines at their site, do not need the extra designation, and use only the designations of *company.domain*. The *proteus* machine is given a domain of *COM* as it is part of the DARPA INTERNET. The two methods that *conam.conam.UUCP* will use to send mail to other machines is via UUCP and Micnet, these methods are known as channels.

Before going any further, it is necessary to present a scenario on how a mail address is deciphered, given an address and the domains thus defined. Suppose the address we wish to send mail to is:

```
dallas@mother.comp.nostromo.COM
```

from *conam.conam.UUCP*. Two things must happen to have this mail delivered properly. The mailer must verify that the host *mother.comp.nostromo.COM* is either known by conam, or that the COM domain is recognized by conam. In either case, once either the host or domain is validated, a path to that host or domain must be found. In other words, once the address is validated, a way of getting the mail to that host or domain must be found. These two tasks (validation and getting it there), are accomplished by two different sets of files, called the domain and channel files. It is necessary to note the distinction of getting a piece of mail to its destination host versus its destination domain. If only a path to a domain is found, then it is the job of the host to which the mail is routed to deliver the mail to its final destination.

NOTE: All files specified in the sections that follow are for *conam.conam.UUCP*.

Domain Files: /usr/mmdf/table/*.dom

Domain files are used to match a host name to its "fully-qualified" domain name. For each domain defined there is a domain file. The domain file can map every name of a host to its "fully qualified" domain name. With the example setup we have domain files *local.dom*, *micnet.dom*, *uucp.dom* and *root.dom* (a special case).

local.dom:

```
conam          conam.conam.UUCP
```

micnet.dom:

```
guardian              guardian.conam.UUCP
colossus              colossus.conam.UUCP
```

uucp.dom:

```
obie           obie.UUCP
```

root.dom:

```
proteus.COM    proteus.COM
COM            proteus.COM
EDU            proteus.COM
MIL            proteus.COM
GOV            proteus.COM
```

We see that the left-hand side is the name of a specific host or domain. The right-hand side is the fully-qualified name of the host, or the name of a host which will be able to route mail on the given domain. root.dom is a special case in that it takes into account all of the hosts and domains not specified elsewhere in the domain files.

Channels

A channel is the method by which the mail will actually be delivered to the appropriate host or domain. For example, UUCP and Micnet are channels. Channel files take fully-qualified names and are used to generate the route to that host or domain.

Channel Files: /usr/mmdf/table/chn.*

A channel is the method by which the mail will actually be delivered to the appropriate host or domain. For us these are UUCP and Micnet, along with the local host's mailer. Channel files take fully-qualified names and tell us what route to take to get to that host or domain. The "%s" means to use the rest of the address at this point.

In our case, we have three channel files, local.chn micnet.chn and uucp.chn.

local.chn:

```
conam.conam.UUCP     conam
```

micnet.chn:

```
guardian.conam.UUCP        guardian:%s
colossus.conam.UUCP        colossus:%s
```

uucp.chn:

```
obie.UUCP     obie!%s
proteus.COM   proteus!%s
```

As you can see, the specific hosts listed above are mapped to their appropriate network addressing scheme, for example "host!user" for UUCP paths and "host:user" for Micnet.

One thing to note here in *uucp.chn* is *proteus.COM*. Although *proteus.COM* is part of the INTERNET, it is connected to *conam* via UUCP, and so we must specify the route to it in UUCP format in *uucp.chn*.

Given the domain and channel files above, the address forbin@cpo.EDU is matched in *root.dom* as:

```
EDU     proteus.COM
```

and in *uucp.chn* as:

```
proteus.COM   proteus!%s
```

and the mail is delivered with the UUCP address of:

```
proteus!%s
```

Aliases

There is also a mechanism for setting up aliases with MMDF. these are the alias files in */usr/mmdf/table* (alias.*). An example is the *alias.user* file, which maps users to machines. This is useful if you want each individual in your organization to receive mail on a particular machine. For example:

```
nathanb    nathanb@guardian
mavrac     mavrac@colossus
ering      ering@obie.UUCP
```

NOTE: If user is in the organization, give only local host name, if user is outside the organization, give fully-qualified domain name.

There is also an alias.list file, for multiple-user aliases. With list aliases you can hide the actual sender's name and have the recipient think it has come from *info-p1*.

```
info-p1             info-p1-outbound@list-processor
info-p1-outbound    ripley@mother.comp.nostromo.COM,ering,nathanb
info-p1-request     nathanb
```

The alias.ali file is for even more aliases, usually these are topic-specific aliases:

```
postmaster    nathanb
toasters      nathanb,wallyb,mavrac
```

The MMDF Configuration File: mmdftailor

Creating the domain, channel and alias files is not enough. The MMDF programs have no idea what is defined unless we specifically tell them what files we have created. This is done with the */usr/mmdf/mmdftailor* file. Figure 16-2 chows the *mmdftailor* file for *conam.conam.UUCP*. Below is the information on how to interpret it:

Four crucial variables are:

MLDOMAIN - the domain of the machine
MLNAME - then name of the company
MLOCMACHINE - the name of this specific machine - same as /etc/systemid
UUname - the name used with uucp - must exist for uucp to work properly

The other keywords that need to be in the mmdftailor file:

MTBL	one entry for each domain, channel and alias file, the MTBL entries tell the MMDF programs what files in */usr/mmdf/table* to use to create the list of valid addresses.
ALIAS	Tells what tables defined with MTBL are alias tables, one for each alias file.
MDMN	Tells what tables defined with MTBL are domain tables, one for each domain file.
MCHN	One entry for each channel, tells what program will actually move the mail in that channel, how often, and in what manner this will occur.

```
;
; First the local domain and system name are defined.  The default
; domain is UUCP.  If an organization only has one machine, you
; could name it unix386.UUCP, for instance.  A number of machines
; could use an organization name such as unix386.company.UUCP.  Or
; a registered domain name is used instead of the default domain UUCP.
MLDOMAIN UUCP
MLNAME   conam
MLOCMACHINE conam
UUNAME conam

; MSUPPORT is the address to which problem notifications concerning
; the delivery of mail are sent.  It _must_ be a valid address.
MSUPPORT postmaster

; The tables:
MTBL     local,          file="local.chn",        show="Local Host Aliases"
MTBL     locdom,         file="local.dom",        show="Local Domain"
MTBL     list,           file="list.chn",         show="List Channel"
MTBL     uudom,          file="uucp.dom",         show="SCO UUCP Domain"
MTBL     uuchn,          file="uucp.chn",         show="SCO UUCP Channel"
MTBL     micdom,         file="micnet.dom",       show="SCO Micnet Domain"
MTBL     micchn,         file="micnet.chn",       show="Micnet Domain"
MTBL     rootdom,        file="root.dom",         show="Root Domain"
MTBL     auser,          file="alias.user",       show="User alias"
MTBL     lalias,         file="alias.list",       show="List Channel Aliases"
MTBL     alias,          file="alias.ali",        show="Local Name Aliases"

;  The aliases:
ALIAS    table=alias,    trusted, nobypass
ALIAS    table=lalias,   trusted, nobypass
ALIAS    table=auser

;  The channels:
MCHN     local, show="Local Delivery", que=local, tbl=local, ap=same,
         pgm=local, mod=imm
MCHN     list, show="List Processing", que=list, tbl=list, ap=733,
         pgm=list, mod=imm
MCHN     micnet, show="SCO UUCP Delivery", que=micnet, tbl=micchn, ap=same,
         pgm=micnet, mod=imm
MCHN     uucp, show="SCO UUCP Delivery", que=uucp, tbl=uuchn, ap=733,
         pgm=uucp, mod=imm
; MCHN   badhosts, show="Last-Chance Routing", que=badhosts, tbl=mnchn,
;        ap=same, pgm=xnet, mod=imm, host=smartmachine.UUCP

;  The domains:
MDMN     "conam.UUCP", show="Local domain", table=locdom
MDMN     "UUCP", show="UUCP Domain", table=uudom
MDMN     "UUCP" show="Micnet Domain", table=micdom
MDMN     "",show="Root Domain", table=rootdom

;  Logging levels:
MMSGLOG  level=FTR,size=20
MCHANLOG level=FTR,size=20
AUTHLOG level=FTR, size=20
```

Figure 16-2. Sample mmdftailor File

Configuring MMDF

MMDF configuration begins with the */usr/mmdf/mmdftailor* file, which defines the local machine and domain names, the various tables (alias, domain, and channel), and other configuration information. The *alias.list* and *alias.user* files contain alias definitions; the *.dom* and *.chn* files define the routing information for each network protocol. The section "Routing Example" demonstrates how MMDF uses the alias and routing tables.

To change the configuration of MMDF on your system, you can log in as **mmdf** and edit the configuration files. Whenever you change MMDF alias or routing information in any way, you must rebuild the hashed database (see "Updating the Database").

This section explains the parts of the **mmdftailor** file and the alias and routing files that you will most likely want to change when setting up your MMDF system. The **mmdftailor**(F) and **tables**(F) manual pages provide complete descriptions of the formats of these files.

Modifying the mmdftailor File

mmdftailor is the top-level configuration file for MMDF. This file contains configuration information and directs MMDF to each of the other configuration files.

Domain and Machine Names

The first few lines of the **mmdftailor** file define the full name of the machine. When you install MMDF with **custom**, these lines are initially set to something like:

```
MLDOMAIN UUCP
MLNAME blue
```

UUCP is a generic domain name, and *blue* is the name of the machine.

For a simple configuration, you can leave your machine name like this. You will want to change these names if:

- you have an officially registered domain name, which allows you to exchange mail through a worldwide network—for information about registering a domain name, write to:

  DDN Network Information Center
  SRI International
  333 Ravenswood Avenue, Room EJ291
  Menlo Park, CA 94025 USA

- you have several machines within your company and you want people outside the company to be able to send mail to anyone without having to know the name of the specific machine inside your company on which the person receives mail

If you have an official domain name (for example, *sco.COM*), you should change the first two lines of **mmdftailor**. If you have several machines within your company, you can add a local machine name to *mmdftailor*. Here is an example:

```
MLDOMAIN COM
MLNAME sco
MLOCMACHINE blue
```

Other than *UUCP*, the most common values for MLDOMAIN are *COM* for commercial organizations and *EDU* for educational organizations. MLNAME specifies the name of your company as it will be known throughout the network. MLOCMACHINE gives the local name of the machine.

Defining MLOCMACHINE allows you to hide the local machine name within your company's registered domain so people sending mail do not have to remember the internal name of a machine. When you join local machines under a single domain name, you create an administrative domain. Within an administrative domain, all user names must be unique so that mail can go to any person anywhere within the domain without a local machine name in the mail address.

In this example, *COM* is the domain, *sco* is the company name, and *blue* is the local machine name. A user on this machine with the name *perry* can receive mail that is simply addressed like this:

perry@sco.COM

Support Address

The next line in the **mmdftailor** file defines the address to which MMDF should send any mail that it cannot deliver or return to its sender. An example is:

```
MSUPPORT postmaster@blue.sco.COM
```

The address you specify with MSUPPORT must be legal; if it is not a legal address and MMDF cannot deliver the original undeliverable mail to the support address, MMDF creates a new piece of mail that is undeliverable and on and on until the machine runs out of processes.

You can designate anyone to receive undeliverable mail, but a local user is best because the address is simpler and therefore more likely to be a valid address.

Delivery Tailoring

If you want MMDF to deliver mail to a file or directory other than the default file named with the user's login in the */usr/spool/mail* directory, you can add lines similar to these:

```
MDLVRDIR ""
MMBXNAME ".mailbox"
MMBXPROT 0600
```

If MDLVRDIR is null, then MMDF delivers to the user's home directory. If MMBXNAME is null, then MMDF uses the user's login as the name of the mailbox file. MMBXPROT sets the protection mode on mailbox files with the same set of octal numbers that the **chmod**(C) command uses to change access permissions. With this example, MMDF delivers to a *.mailbox* file in the user's home directory with a file protection that ensures that only the owner can read or write to the file.

Table Definitions

The next section of the **mmdftailor** file defines the alias, domain, and channel tables. Each line associates an abbreviated name and a more descriptive name with the file containing the table, which is located in the directory */usr/mmdf/table*. The abbreviated names are used later in this file as a shorthand to refer to the table files. The more descriptive name is used by certain programs as a display line to explain what the table is.

For example, the alias table of user-to-machine mappings can be defined as:

```
MTBL auser, file="alias.user", show="User Aliases"
```

The file */usr/mmdf/table/alias.user* can now be referred to as *auser* throughout the rest of the **mmdftailor** file.

Although you will probably not change the existing table definitions, you should know how each table is defined as you modify other parts of the **mmdftailor** file. If you set up a new channel, the network package's installation script should add the appropriate table definitions to **mmdftailor**.

Alias Definitions

The ALIAS entries define the various sources of alias information, using the abbreviated names specified in the MTBL definitions. Each alias table can be defined with these characteristics:

trusted — A trusted alias file can direct mail to be delivered to any file or process using the permissions of any user on the system (including *root*); only the super user should have access to modify a trusted alias file.

nobypass — This option prevents the *~address* alias bypass mechanism from working on the aliases in this file.

Here are some sample alias definitions:

```
ALIAS table=lalias, trusted, nobypass
ALIAS table=auser
```

MMDF searches the alias tables in the order you list them, using the first alias that matches without looking for other matches in later tables. The section "Defining Aliases" describes how to create the alias files.

Channel Definitions

The MCHN entries define the channels available to MMDF for mail transport. A channel is the mechanism for delivering mail either to a mailbox on the local machine or across the network to a remote machine.

At least two channels are required: one for delivering local mail and one for processing large mailing lists (the **list**(ADM) manual page explains how mailing lists are handled). You should define other channels for the network protocols you want configured into your system.

Channel definitions look like this:

```
MCHN   local, show="Local Delivery", que=local,
       tbl=local, ap=same, pgm=local, mod=imm
MCHN   list, show="List Processing", que=list,
       tbl=list, ap=same, pgm=list, mod=imm,
       host="sco.COM", confstr=sender
MCHN   uucp, show="UUCP Delivery", que=uucp,
       tbl=uuchn, ap=822, pgm=uucp, mod=imm
MCHN   micnet, show="Micnet Delivery", que=micnet,
       tbl=mnchn, ap=same, pgm=micnet, mod=imm
MCHN   badhosts, show="last-chance routing",
       que=badhosts, tbl=mnchn, ap=same, pgm=micnet,
       mod=imm, host="sco.sco.COM"
```


The order of the MCHN definitions is important because MMDF searches the channel tables in this order.

The last channel defined in the example (*badhosts*) is used for mail addressed to a host that the **submit**(ADM) program does not recognize. This channel forwards the mail to a host that has a better host database. *badhosts* is not really a channel because it is not associated with its own transport program; this pseudo-channel uses the Micnet channel to relay mail to a more intelligent host. If the *badhosts* channel does not exist, mail to an unknown host is returned to its sender.

In the channel definitions, the first argument is the name of the channel. The parameters used to define these channels are:

show — gives a descriptive name used by certain programs as a display line to explain the channel's function

que — specifies the subdirectory of */usr/spool/mmdf/lock/home* in which to queue messages for this channel; the name given here is prefixed with "q." to form the subdirectory name (see the **queue**(F) manual page for more detail)

tbl — uses the abbreviated name from the MTBL definition to specify the channel table

ap — selects the type of address parsing used for the header of outgoing messages

822	converts to RFC822-style addresses
same	does not reformat headers

pgm — indicates the program in the */usr/mmdf/chans* directory that takes mail from the **deliver**(ADM) program and carries it to its destination on the local machine or across the network to a remote machine.

mod — sets the delivery mode for the channel

imm	sends mail immediately
reg	queues mail, but does not send it; you must run **deliver** to actually send mail through a regulated channel (this is the default)

host — names the intelligent host to which a channel relays all its mail; the list channel must have this parameter set to the local host

confstr — passes a channel-specific flag to the program run by the channel; the list channel uses a configuration string to enable **sender** mode, so that if no *list-request* alias is defined for a mailing list, the sender of the message is recorded as the person who mailed to the list (instead of recording the postmaster as the sender)

See "Editing the Routing Files" to learn about the contents and function of channel files.

Domain Definitions

The MDMN entries define the domains known to MMDF. A domain is a collection of machines that are related in some way, possibly by geographic location (*CAMFORD.AC.UK*), organization (*sco.COM*), or type of activity (*OXBRIDGE.EDU*). Domain definitions look like this:

```
MDMN "sco.COM", show="Local Domain", table=locdom
MDMN "UUCP", show="UUCP Domain", table=uudom
MDMN "LIST", show="List Pseudo-Domain", table=list
MDMN "", show="Root Domain", table=rootdom
```

The first argument is the name of the domain. The root domain definition has no name ("") because the root domain table can contain entries for many different domains.

The *show* parameter gives the domain a more descriptive name for use as a display line by certain programs. The *table* parameter uses the abbreviated name from the MTBL definition to specify the domain table.

The list domain handles mail sent to a large mailing list by verifying the addresses in the background to speed up processing for the person sending the mail (see the **list**(ADM) manual page). *LIST* is not really a domain because it is not associated with a collection of machines; this pseudo-domain uses the list channel to expand the mailing list and remail the individual messages.

MMDF searches for the longest possible match for a domain. For example, for mail addressed to *CAMFORD.AC.UK*, the *AC.UK* domain table is matched before the *UK* domain table. If MMDF cannot find an exact match, it looks for a partial match and routes the mail in that direction. For example, if mail is addressed simply to *CAMFORD* and no *CAMFORD* domain table exists, MMDF searches the domain tables in the order you listed them for a *CAMFORD* entry. MMDF routes the mail to the domain in which it finds a partial match.

When MMDF cannot find even a partial match in earlier domains, it looks for a match in the root domain to send the mail on to a more intelligent host. When MMDF finds no match at all, as a last resort, it uses the *badhosts* channel, if it exists. Because MMDF uses the first domain that has an exact match without looking for other matches in later tables, the order in which you list MDMN definitions is significant. Be sure the local domain is first and the root domain is last.

See "Editing the Routing Files" for the contents and function of domain files.

Logging Levels

The last section of the **mmdftailor** file sets the level of information to be saved and the maximum size for the MMDF log files, which are stored in the */usr/mmdf/log* directory. For example:

```
MMSGLOG level=FAT, size=20
```

MMSGLOG controls the log file *msg.log*, which is produced by the **deliver** and **submit** programs; AUTHLOG controls the authorization information saved in *auth.log*; MCHANLOG controls the logging of most other MMDF programs in the file *chan.log*.

The most verbose levels of logging produce enormous amounts of data and slow down processing. The common settings for the *level* parameter are (in order of increasing detail):

FAT	logs fatal errors only
GEN	saves the generally interesting diagnostics
BST	shows basic statistics
FST	gives full statistics

With the *size* parameter, you can limit the size of the log file by setting the number of 25-block units that the file is allowed to grow to. For the MMSGLOG example, fatal errors only are logged until the file reaches 500 blocks (20 times 25). When the log file reaches the specified size, logging stops. You should periodically check the log files for errors and clean out the files before they reach the maximum size.

The **logs**(F) manual page gives more detail about the MMDF log files.

Defining Aliases

The MTBL definitions in the */usr/mmdf/mmdftailor* file direct MMDF to the *alias.list* and *alias.user* files in the */usr/mmdf/table* directory for alias definitions. You can create and edit these files as described in this section or by studying the file syntax described in the **tables**(F) manual page. Whenever you change the *alias.list* or *alias.user* file in any way, you must rebuild the hashed database.

alias.list File

The *alias.list* file contains list-type aliases, which assign a single name to represent:

- one or more user names or other aliases
- redirection of the message to a file
- redirection of the message to a pipe
- a mailing list

For example:

```
postmaster:          admin, perry, Loguucp
Loguucp:             "network//usr/spool/log/uucp"
Logmlog:             "network|cat -v >>/usr/spool/log/mlog"
printer2:            "network|/usr/bin/lpr -dprinter2"
staff:               staff-outbound@list-processor
staff-outbound:      ":include:/etc/alias/staff"
staff-request:       ross
```

You should designate a local user as the postmaster for your system and define a *postmaster* alias. In this example, mail addressed to *postmaster* goes to the users *admin* and *perry* and is recorded in the UUCP log file. The slash syntax for redirection is useful for recording activity directly in a log file.

You can also use the normal output redirection symbol (>) with pipe redirection to do more complex processing. Mail addressed to *Logmlog* is piped to the **cat**(C) command, then logged in the file *mlog*. Mail addressed to *printer2* is piped to the **lpr**(C) command for printing. The redirection aliases use the user and group IDs of the user *network*. Although *network* is appropriate in most cases, you can specify any user named in the */etc/passwd* file.

The last three lines handle the processing of the *staff* mailing list. This example shows how the ":include:" syntax uses the names listed in the specified file to define the alias. You can also use the normal input redirection symbol (<) to read an alias definition from a file. The **list**(ADM) manual page explains in detail how to set up mailing lists.

In the *alias.list* file, the alias name and its definition can be separated by whitespace, a colon, or both. When defining an alias that contains many user names, you can use a backslash (\) as a line continuation character. Use quotation marks (" ") to delimit a string containing spaces or punctuation. When using an alias to define another alias, be careful not to create an alias loop.

alias.user File

The *alias.user* file contains aliases that map users to their home machines. For example:

```
admin:               admin@blue
carmen:              carmen@ivy
perry:               perry@blue
ross:                ross@warwick
```

Editing the Routing Files

MMDF routing is controlled by the domain (*.dom*) and channel (*.chn*) files. A domain file entry maps a machine name (*blue*) to a fully qualified domain name (*blue.sco.COM*), which is the first host to which the mail should go to start on its way to its destination. (In many cases, this host is the destination.) A channel file entry maps a host to the transport address that delivers to that host.

You can create and edit the domain and channel files as described in this section or by studying the file syntax described in the **tables**(F) manual page. Whenever you change a *.dom* or *.chn* file in any way, you must rebuild the hashed database.

Domain Files

The MDMN definitions in the */usr/mmdf/mmdftailor* file direct MMDF to search the specified *.dom* files in the */usr/mmdf/table* directory for domain definitions.

The first domain defined in */usr/mmdf/mmdftailor* should be the local domain. The *local.dom* file contains an entry for each machine within the local domain. Each entry expands the local machine name on the LHS (left-hand side) to the fully qualified domain name on the RHS (right-hand side). A *local.dom* file might look like this:

```
blue                    blue.sco.COM
ivy                     ivy.sco.COM
warwick                 warwick.sco.COM
```

In addition to a local domain file, you will probably also have a UUCP domain file (*uucp.dom*). In this file, you can list the machines within the *UUCP* domain that you mail to frequently. Each entry expands an abbreviated or alternate name on the LHS to the UUCP host name on the RHS. For example:

```
mcvax                   mcvax.UUCP
vu44                    vu44.UUCP
```

Any *UUCP* machine not listed in this domain is handled by default routing through the *UUCP* channel.

If you have a */usr/lib/uucp/Systems* file, you can initially create the *uucp.dom* file by converting the *Systems* file with the **uulist** conversion script (see "Configuring for UUCP").

If you have a XENIX-format Micnet topology file (*/usr/lib/mail/top*), you can initially create the *micnet.dom* file by converting the *top* file with the **mnlist** conversion script (see "Configuring for Micnet").

Following this pattern of the abbreviated name on the LHS mapped to the host name on the RHS, you can create a domain file for each MDMN definition in **mmdftailor** (except the list pseudo-domain, which uses the local domain file). In these *.dom* file, the RHS is formed by appending the name of the domain (as defined in the MDMN definition) to the LHS. The LHS and RHS can be separated by whitespace, a colon, or both. The last domain defined in */usr/mmdf/mmdftailor* should be the root domain. This special domain file (*root.dom*) maps a domain name on the LHS to a host name on the RHS. The *root.dom* file can contain entries that specify:

- a path to a particular domain that is not included in another domain table
- a more intelligent host to which to send mail that is addressed to a machine that the local machine does not recognize

Here are examples of these types of *root.dom* entries:

```
sri-nic.arpa                    sri-nic.arpa berkeley.EDU
com                             uunet.UU.NET
```

If *sri-nic.arpa* were the only host in the *arpa* domain that you access, you would probably not want to create a separate domain file for *arpa*. Instead, the first entry routes mail addressed to *sri-nic.arpa* through *berkeley.EDU*.

This example also shows how you can specify a path through which to access a machine that is not directly accessible to the local machine. The path on the RHS is read from right to left and can contain several intermediate hosts. The host furthest to the right must make a direct connection to the local host.

Because the root domain is searched last, the *root.dom* file can contain a top-level domain name (such as *COM*) that is used if a domain name is not matched more specifically in an earlier domain. If mail is addressed to *ross@nesser.COM* and *nesser.COM* is not matched at all in any domain file, then the top-level *COM* domain would match the second entry, and MMDF would pass this mail on to *uunet.UU.NET*, with the hope that *uunet.UU.NET* knows how to get the mail to *nesser.COM*.

Channel Files

The MCHN definitions in the */usr/mmdf/mmdftailor* file direct MMDF to search the specified *.chn* files in the */usr/mmdf/table* directory for channel definitions.

The *local.chn* file contains entries like this:

```
sco.COM                 sco
sco                     sco
blue.sco.COM            sco
blue                    sco
```

You must include the first two entries that map *MLNAME.MLDOMAIN* and *MLNAME* to *MLNAME* as defined in the **mmdftailor** file. If you are hiding local machines, you must also include the last two entries that map *MLOCMACHINE.MLNAME.MLDOMAIN* and *MLOCMACHINE* to *MLNAME*.

The *list.chn* file contains:

```
list-processor          list-processor
list-proc               list-processor
```

The LHS is a pseudo-host defined in a mailing list alias (see the sample *alias.list* file). These entries tell MMDF to pass mail addressed to a mailing list to the list-processor program.

The *uucp.chn* file contains entries like this:

```
mcvax.uucp              uunet!mcvax!%s
sri-nic.arpa            uunet!sri-nic.arpa!%s
uunet.uu.net            uunet!%s
```

The LHS is the UUCP host name; the RHS is the UUCP address that MMDF uses when it invokes the UUCP software. With the first entry in this example, when mail is addressed to the user *hillis* at *mcvax.uucp*, the UUCP channel passes the mail to *uunet* along with the rest of the UUCP address (*mcvax!hillis*). The second entry shows how a domain name (*sri-nic!arpa*) can be used within a UUCP path.

The Micnet channel file (*micnet.chn*) contains entries like this:

```
ivy.sco.COM             ivy:%s
warwick.sco.COM         ivy:warwick:%s
```

The LHS is the host name from the *local.dom* file; the RHS is the Micnet address that MMDF uses when it invokes the Micnet software. With this example, when mail is addressed to the user *ross* (who receives mail on the machine *warwick*), the Micnet channel passes the mail to *ivy* along with the rest of the Micnet address (*warwick:ross*).

Following this pattern of the host name on the LHS mapped to the addressing information for delivering to that host on the RHS, you can create a channel file for each MCHN definition in **mmdftailor** (except the *badhosts* pseudo-channel, which uses the Micnet channel file). The LHS and RHS can be separated by whitespace, a colon, or both.

Changing Your Machine or Site Name

During installation, you were prompted to provide a name for your machine. To change your system name after installation, you must log in as **mmdf** and do the following:

1. Move to the **mmdf** directory:

   cd /usr/mmdf

2. Edit the *mmdftailor* file and alter the MLNAME and UUname entries at the top of the file to reflect the desired name change. In addition, any other occurrences of the old system name in this file should be changed to reflect the new name.

3. Edit any **.dom* and **.chn* files in the */usr/mmdf/table* directory and change all instances of the old system name to reflect the new system name.

4. Enter the following commands:

   cd /usr/mmdf/table
   ./dbmbuild

Your system will then use the new name.

Routing Example

When mail is addressed to *postmaster*, MMDF routes the mail by first searching the hashed alias table from *alias.list* to expand the *postmaster* alias to the associated user names. An entry in the *alias.list* file might be:

```
postmaster:          perry
```

Then, MMDF searches the *alias.user* file to find the local machine name associated with user name. The *alias.user* file might contain:

```
perry:               perry@blue
```

MMDF searches the various *.dom* files, which map the local machine name to a fully qualified domain name. In this case, the machine *blue* is in the local domain so MMDF finds the following entry in *local.dom*:

```
blue                 blue.sco.COM
```

MMDF then searches the various *.chn* files, which map the fully qualified domain name to addressing data. In this case, the domain *blue.sco.COM* is served by the local channel so MMDF finds the following entry in *local.chn*:

```
blue.sco.COM            sco
```

According to the MCHN definition in **mmdftailor**, the local channel queues mail in the file */usr/spool/mmdf/lock/home/q.local*, and the program **/usr/mmdf/chans/local** delivers the mail to Perry's mailbox.

Updating the Database

The hashed database gives MMDF quick access to alias and routing information. You must update this database whenever any alias or routing file changes. To rebuild the database, log in as **mmdf** and run the program **/usr/mmdf/table/dbmbuild** from the */usr/mmdf/table* directory:

```
cd /usr/mmdf/table
dbmbuild
```

The **dbmbuild** program uses the definitions in the **mmdftailor** file to build the hashed database and reports if any tables are missing. See the **dbmbuild**(ADM) manual page for full details.

Maintaining Your MMDF System

The **cleanque** program cleans the mail queues of outdated files. Arrange for **cron**(C) to run **cleanque** at least daily (maybe even more often, depending on your mail volume). You can also run **cleanque** by hand whenever you suspect a problem with mail delivery. The **cleanque**(ADM) manual page provides a complete description of this program.

The **checkque** program checks the status of the mail queues and shows how many messages are waiting for delivery. If a queue is backed up with waiting mail, you can try delivering the mail manually with the **deliver**(ADM) program:

```
deliver -w -clist,uucp
```

The **-c** option specifies the channels to be processed. The **-w** option causes **deliver** and the channel programs to output informative messages as they try to deliver the mail. (**deliver** is

discussed below.) You can review the output for abnormalities, like a rejected sender or recipient. The **checkque**(ADM) manual page provides a complete description of this program.

deliver(ADM) controls delivery of messages on a per-channel basis. The options are:

-b run in background

-T*n* attempt delivery after every *n* seconds

-c*chan* deliver messages on specified channel

-w generate debugging information

-L*file* use file as logfile instead of /usr/mmdf/log/chan.log

There can be several **deliver** daemons; the following are some examples:

```
su mmdf -c "/usr/mmdf/bin/deliver -clocal,uucp,badhosts,list -b"
su mmdf -c "/usr/mmdf/bin/deliver -b -T60 -clocal -L/usr/mmdf/local.log"
su mmdf -c "/usr/mmdf/bin/deliver -b -T3600 -cuucp -L/usr/mmdf/log/uucp.log"
```

Such daemons can be placed in a file named */etc/rc2.d/S86mmdf*. (See **deliver**(ADM) for more information.)

Converting Existing Configuration Files

To help configure your MMDF system, conversion utilities are provided to produce MMDF-compatible alias and routing files from XENIX-format files. With these utilities, stand-alone and simple Micnet and UUCP configurations can be created without any hand editing of the MMDF configuration files. To configure a more complex system, you will need to edit the MMDF configuration files by hand.

Before you can begin any conversion, you must restore the XENIX alias and routing files from backup tape. After installing MMDF with **custom**, restore these files:

/usr/lib/mail/aliases
/usr/lib/mail/top
/usr/lib/uucp/Systems

The next sections demonstrate how to convert these XENIX files to MMDF format. If you do not have these XENIX-format alias and routing files, you must create the MMDF-format alias and routing tables by hand (see "Configuring MMDF").

Whenever you change MMDF alias or routing information in any way, you must rebuild the hashed database (see "Updating the Database").

Converting an Alias File

The **mmdfalias** utility changes aliases in the file */usr/lib/mail/aliases* from the XENIX format:

machine?user

to the MMDF format:

user@machine

For example, *blue?perry* becomes *perry@blue*.

mmdfalias also splits the converted contents of the XENIX file into two MMDF files containing list-type aliases and aliases that map users to machines. To ensure that the XENIX aliases file is split correctly, before starting the conversion, edit the */usr/lib/mail/aliases* file that you backed up from tape to add the following comment line as a separator between the list-type aliases and the user-to-machine aliases. Ensure that the list aliases are before the separator and that the user-to-machine mappings follow it.

```
# user-to-machine mapping
```

Then, to convert the XENIX aliases file to MMDF format, log in as **mmdf** and run the **/usr/mmdf/table/tools/mmdfalias** conversion script from the */usr/mmdf/table* directory:

cd /usr/mmdf/table
tools/mmdfalias

mmdfalias creates two new files, *alias.list* and *alias.user*, in the current directory (in this case, */usr/mmdf/table*). These two files must exist in */usr/mmdf/table* before you update the database.

Configuring for Micnet

If you want to be able to route mail over Micnet, you must build MMDF domain and channel files from the Micnet topology file */usr/lib/mail/top*. First, make sure that the Micnet

topology has been built properly, using the program **netutil**. The file */usr/lib/mail/top* should contain an entry for each pair of machines connected via micnet, like:

```
black    tty1a    blue    tty5d    9600
black    tty1a    red     tty5e    9600
```

This would indicate that the machines *black* and *blue* are connected together, as are *black* and *red*.

Now, log in as the user **mmdf** and use the script **mnlist**(ADM) to build the MMDF files *micnet.dom* and *micnet.chn*:

cd /usr/mmdf/table
tools/mnlist

To be sure that the domain and channel files were built properly, look at the files *micnet.dom* and *micnet.chn* and see that they contain entries for each of the machines in the Micnet, like:

```
micnet.dom:
    black          black.UUCP
    blue           blue.UUCP
    red            red.UUCP

micnet.chn:
    black.UUCP     black:%s
    blue.UUCP      blue:%s
    red.UUCP       red:%s
```

Note that the domain ".UUCP" is assumed for the machines in the Micnet. To change this, either the micnet.dom and micnet.chn files may be edited by hand after running **mnlist**, or the script **mnlist**(ADM) may be edited and the line "LDOMAIN=UUCP" changed to reflect the local domain, and then **mnlist** may be run to generate domain and channel files with the proper domain.

Once the domain and channel files have been generated and found to be accurate, while still logged in as *mmdf* and within the directory */usr/mmdf/table*, rebuild the database:

dbmbuild

Configuring for UUCP

If you want to be able to route mail over UUCP, you must build MMDF domain and channel files from the */usr/lib/uucp/Systems* file, which contains the sites your machine is permitted to contact, as in the following example:

```
obie Any ACU 1200 4444444 ogin:-BREAK-ogin:-BREAK-ogin: \
  uubig word: wetrot
mavra Any1800-0700 ACU 2400 18888888 "" \r ogin:-BREAK-ogin: \
  -BREAK-ogin:nuucp
uunet Any1800-0700 ACU 2400 17031111111 ogin:-BREAK-ogin: \
  -BREAK-ogin:xytpq sword: grm5q
```

Now, log in as the user **mmdf** and use the script **uulist** to build the MMDF files *uucp.dom* and *uucp.chn*:

cd /usr/mmdf/table
tools/uulist

To be sure that the domain and channel files were built properly, look at the files *uucp.dom* and *uucp.chn* and see that they contain entries for each of the machines in the UUCP network, like:

```
uucp.dom:
    obie          obie.UUCP
    mavra         mavra.UUCP
    uunet         uunet.UUCP

uucp.chn:
    obie.UUCP     obie:%s
    mavra.UUCP    mavra:%s
    uunet.UUCP    uunet:%s
```

Once the domain and channel files have been generated and found to be accurate, while still logged in as *mmdf* and with the current directory still **/usr/mmdf/table** rebuild the database:

dbmbuild

Chapter 17
Adding Hard Disks

When your system suffers from chronic lack of space, you probably need to add a hard disk to give the system extra space for storing user files and directories. The following types of secondary hard disks and controllers are supported:

- ST506 (IBM AT standard)
- ESDI (SMS OMTI 8620 or 8627 controller)
- SCSI

Three configurations are possible:

- Root disk on a SCSI host adapter with an option of adding one additional SCSI host adapter, each supporting up to seven controllers, and each SCSI controller supporting up to eight devices.
- Root disk on an ST506 controller with an option of adding one additional ST506 controller, each supporting two ST506 disks and up to two SCSI host adapters (which can be configured as in the first option).
- Root disk on an OMTI controller that supports an additional disk, both of which can be either ESDI or ST506.

Figure 17-1 illustrates a configuration of the second type.

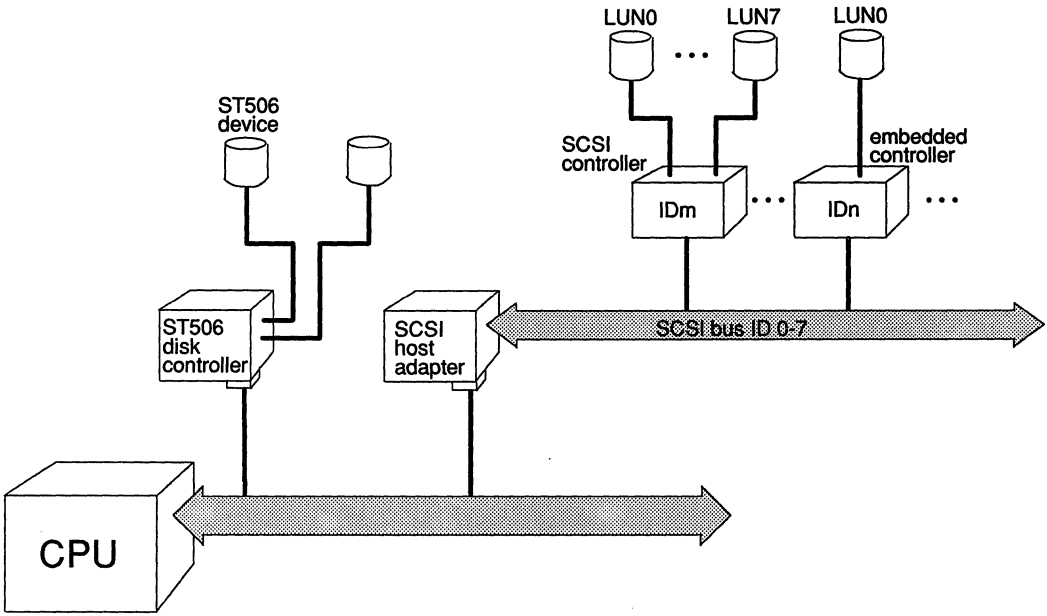


Figure 17-1. ST506 and SCSI Configuration Examples

A SCSI host adapter (HA) translates signals from the CPU bus to the SCSI bus. A SCSI controller is known as a SCSI ID. A SCSI device is referenced by a logical unit number (LUN).

When you initialized your root disk as you installed the operating system, the root disk was configured as the first hard disk on the first controller (for ST506 or ESDI disks) or first host adapter (for SCSI disks).

Although the basic procedure for adding a disk is common to all types of disks, you will occasionally need to perform somewhat different steps based on the type of disk you are installing. Throughout the procedure, the differing steps are clearly indicated.

Before You Start

Before installing an additional hard disk, you must first decide how to configure the disk, then set up and connect the hardware. This section explains the syntax used for the **mkdev hd** command used to configure and add hard disks. You should use this section to determine what command line options are necessary to configure your ST506 and ESDI disks. When you have chosen the proper syntax, you can proceed to "Installing the Hard Disk." In the case of SCSI disks, you must read this section, invoke **mkdev hd** command as instructed, then proceed to "Installing the Hard Disk," where you will invoke the same command a *second* time. This is necessary because the SCSI configuration files must be prepared in the first pass and the disk initialized in the second.

Configuring a Hard Disk

You need to decide how you will configure the disk so you can provide that information to the installation utility.

ST506 or ESDI Disk

To configure an ST506 or ESDI disk with the **mkdev hd** command, you must know which disk controller will support the new disk and whether it is the first or second disk on the controller.

The command syntax is:

mkdev hd *disk controller*

Numbering of disks and controllers starts at 0.

Table 17.1.

ST506 and ESDI Commands

Controller	Command	Disk being added
ST506 & ESDI	**mkdev hd 0 0**	first disk on first controller (root)
	mkdev hd 1 0	second disk on first controller
ST506 ONLY	**mkdev hd 0 1**	first disk on second controller
	mkdev hd 1 1	second disk on second controller

SCSI Disk

To configure a SCSI device with the **mkdev hd** command, you must know:

- the ID number of the controller (0-7) on the host adapter; the host adapter itself is usually ID 7, giving it the highest priority on the SCSI bus
- the host adapter number (SCSI-0 or SCSI-1)
- the logical unit number of the device (0-7) on the SCSI ID; on an embedded controller (where the controller and the device are one physical unit), the LUN is usually 0

Refer to Figure 17-1 for a pictorial representation of each value. You can choose to be prompted for these values, or you can provide them on the command line. Use the following syntax to specify the configuration:

mkdev hd *id host lun*

where:

id is a number from 0-7

host is "SCSI-0" or "SCSI-1"

lun is a number from 0-7

The instructions that follow assume that the **mkdev hd** command is invoked without arguments. If you do provide all the information on the command line, you can skip to the last step of the procedure that follows. You must provide the identical arguments when you invoke **mkdev hd** the second time.

To add your SCSI disk, follow these steps:

1. Enter the following command:

   mkdev hd

   Δ **sysadmsh** users select: System→Hardware→HardDisk

2. **If your root disk is attached to an ST506 controller**, you see the following:

```
Your root hard disk is attached to an ST506 controller.

Pick one of the choices below or you may quit and
invoke mkdev hd -u for a detailed usage message.

                1) Add a hard disk to ST506 controller
                2) Add a hard disk to SCSI controller

Enter 1, 2 or enter q to quit:
```

Enter **2** and press **Return**.

3. **If your root disk is attached to a SCSI controller**, you see the following:

```
Your root disk is attached to a SCSI controller.

The only available choice is to add another SCSI disk.
Do you want to add another SCSI disk?
(For detailed usage message, pick 'n' to
exit this script and invoke mkdev hd -u) (y/n)
```

Enter **y** and press **Return**.

4. Next you see:

```
What is the ID of the controller for this device?
Select 0-7, or 'h' for help, or 'q' to quit:
```

Enter the number of controller attached to the adapter.

5. Next, you enter the number of the adapter that the disk is attached to:

```
Which SCSI host adapter supports this device?
Select 0 or 1, or 'h' for help, or 'q' to quit:
```

Enter **0** if it is the first host adapter, or **1** for the second host adapter.

6. You are then prompted:

```
What is the LUN of this device?
Select 0-7, or 'h' for help, or 'q' to quit:
```

   Enter the number of the device attached to the controller. With most disks, the controller and the device are a single unit, in which case the Logical Unit Number is 0.

7. Now that the data has been entered, the program acknowledges the information and prompts for relinking of the kernel:

```
Updating SCSI configuration file...
The SCSI configuration file has been updated.

A new kernel must be built, to reflect the changes
to the SCSI configuration. Do you want to do this now? (y/n)
```

   The kernel must be reconfigured to recognize the new disk. You are given the option of not relinking in case you are adding a number of devices. This way the kernel need be relinked only once.

8. After the reconfiguration is complete, the following message is displayed:

```
After the system is rebooted with the new kernel,
reinvoke mkdev hd to initialize the new SCSI hard disk.
Use the same values for Host Adapter, ID, and LUN.
```

You now have configured the necessary software support for your new disk. You can now proceed to "Preparing the Hardware."

Preparing the Hardware

Hard disks that do not have matching entries in the ROM tables are supported through software. When adding secondary hardware, you must change some of the switch settings on the host adapter, SCSI ID, and disk. "SCSI Guidelines" in the *Release Notes* explains what these settings should be. Check the hardware manual for your hard disk drive and the computer for instructions.

When you change the settings on a SCSI device with an embedded controller, remember to use the SCSI ID number, not the LUN. The LUN on an embedded controller is 0 since it is the first and only device on the controller.

Before adding the new disk, you must know how to connect it to the computer. Connecting the hard disk is explained in the hardware manual provided with the disk.

Make sure the additional drive is formatted and passes the manufacturer diagnostics before installing the system. If it does not pass the diagnostic tests, you cannot use it with your system.

Installing the Hard Disk

These are the steps to install another hard disk with one UNIX filesystem and no DOS area:

1. After you have connected the hard disk and booted the system, enter system maintenance mode and use the appropriate form of the **mkdev** command, specifying the required configuration information on the command line:

   mkdev hd *disk controller*

   Δ **sysadmsh** users select: System→Hardware→HardDisk

2. **If your root disk is attached to an ST506 controller**, you see the following:

```
Your root hard disk is attached to an ST506 controller.

Pick one of the choices below or you may quit and
invoke mkdev hd -u for a detailed usage message.

                1) Add a hard disk to ST506 controller
                2) Add a hard disk to SCSI controller

Enter 1, 2 or enter q to quit:
```

Enter a number and press **Return**. If you are adding an ST506 disk, proceed to Step 6. If you are adding a SCSI disk, proceed to Step 5.

3. **If your root disk is attached to an OMTI controller**, you see the following:

```
Your root hard disk is attached to an OMTI controller.

The only available choice is to add one other hard disk
Enter 'y' to add another disk.  If you enter 'n' you will exit
this script.  You may then invoke mkdev hd -u for a detailed
usage message. (y/n)
```

Enter **y** and press **Return**. Proceed to Step 6.

4. **If your root disk is attached to an SCSI controller**, you see the following:

```
Your root disk is attached to a SCSI controller.

The only available choice is to add another SCSI disk.
Do you want to add another SCSI disk?
(For detailed usage message, pick 'n' to
exit this script and invoke mkdev hd -u) (y/n)
```

Enter **y** and press **Return**.

5. You are then prompted to enter the same controller, host adapter and LUN information you provided earlier:

```
What is the ID of the controller for this device?
Select 0-7, or 'h' for help, or 'q' to quit:

Which SCSI host adapter supports this device?
Select 0 or 1, or 'h' for help, or 'q' to quit:

What is the LUN of this device?
Select 0-7, or 'h' for help, or 'q' to quit:
```

Enter each as instructed and procced to Step 7.

6. **If you are adding an ST506 or ESDI disk**, you see the following:

```
Will this disk be the first or second disk on this controller?"
Enter 1 (first) or 2 (second):
```

Enter a number and press **Return**.

7. **If you are adding an ST506 disk only**, you see the following:

```
Will this disk attach to the first or second ST506 controller?
Enter 1 (first) or 2 (second):
```

Enter a number and press **Return**.

8. The following prompt is displayed:

```
During installation you may choose to overwrite all
or part of the present contents of your hard disk.
Do you wish to continue? (y/n)
```

Enter **y** and press **Return**.

9. **If you have a SCSI controller,** you see the following message:

```
The hard disk installation program will now invoke /etc/fdisk.
Entering 'q' at the following menu will exit /etc/fdisk.
and the hard disk installation will continue.

If you wish to exit the entire installation at this menu,
press the <DEL> key.
```

Skip to Step 15.

NOTE: The SCSI installation skips Steps 10-14

10. **If you have an ST506 (standard interface) controller,** you see the following message and prompt:

```
The hard disk installation will now invoke /etc/dkinit.
Entering 'q' at the following menu will exit /etc/dkinit,
and the hard disk installation will continue.

If you wish to exit the entire installation at this menu,
press the <DEL> key.

Hard Disk Drive 1 Configuration

        1. Display current disk parameters
        2. Modify current disk parameters
        3. Select default disk parameters

Enter an option or 'q' to quit:
```

11. **If you have an OMTI controller,** you see the following additional message:

```
Caution: Consult the ESDI installation Release Notes if
you wish to modify the disk parameters the /etc/default
will display.
```

Read the section "ESDI Guidelines" in your *Release Notes.*

If you enter **q**, you see the following message:

```
The hard disk installation program will now invoke
two disk preparation utilities:  fdisk and badtrk.
Selecting 'q' at the main menu for each utility
will exit that utility and continue with the hard
disk installation.
```

Skip to Step 15.

12. The **dkinit** menu is intended for unusual or non-standard disks. If you have a standard hard disk, one that is supported by your computer hardware or special motherboard ROM, enter **3** followed by **Return** to continue the installation. In addition, if your disk is a SCSI, you must also enter **q**; the parameters are already set.

    Entering **q** at this point selects the default parameters for your hard disk. Unless you know that your disk is non-standard, assume that it is standard and enter **q** to continue your installation. Skip to Step 16.

    If your disk is non-standard, you must enter in information that overrides the ROM disk configuration information, replacing it with the new information. If you are unsure of what parameters to enter for your non-standard disk, contact your disk manufacturer for this information. The **dkinit** program (called during installation) uses parameters as defined in the "Fixed Disk BIOS Parameter Table" in Section 5 of the IBM Technical Reference (AT).

    If you enter **1** or **2**, you see the following display:

```
Disk Parameters    Values
1. Cylinders       value
2. Heads           value
3. Write Reduce    value
4. Write Precomp   value
5. Ecc             value
6. Control         value
7. Landing Zone    value
8. Sectors/track   value
```

    In the actual display, *value* is replaced with the default value for that variable.

NOTE: The "Cylinders" value refers to the number of cylinders on the entire hard disk and should not to be confused with the number of cylinders you allocated (or intend to allocate) to a given partition.

If you entered a **1**, you now see the first menu again. If you entered a **2**, you are now prompted:

```
Enter a parameter to modify or 'q' to return to the main menu:
```

Enter any of **1** - **8** to change the disk parameters, or **q** to return to the previous menu.

13. You see the following:

```
Enter the new value or <RETURN> to use the existing value:
```

If you wish to change the value, enter a new value now or press **Return** to use the existing value.

14. After you finish changing the disk parameters, enter **q** to return to the main menu. Next, enter **q** again to save the changes you made. Exiting from **dkinit** by entering q overwrites any parameters you have changed with the new values. If you wish to restore the default parameters after making modifications, enter **3** from the first menu.

15. The installation program next runs the **fdisk**(ADM) utility to partition the hard disk. You can partition your disk to also support DOS on the same hard disk (if you have DOS already installed), or you can use the whole disk for your UNIX system.

After a moment, the **fdisk** menu appears on the screen. You see this option list:

```
        1. Display Partition Table
        2. Use Entire Disk for UNIX
        3. Use Rest of Disk for UNIX
        4. Create UNIX Partition
        5. Activate Partition
        6. Delete Partition

Enter your choice or 'q' to quit:
```

Select option **1** and press **Return**.

If you have never installed an operating system on your disk, you see a table similar to this:

```
Current Hard Disk Drive: /dev/rhd10
```

Partition	Status	Type	Start	End	Size

```
Total disk size: 1220 tracks (5 reserved for masterboot
                                          and diagnostics)
Press <RETURN> to continue
```

If you have previously installed an operating system on your disk, the **fdisk** table is filled in. DOS is usually displayed as partition number 4.

16. Press **Return** to return to the main **fdisk** menu. If you would like the UNIX partition to occupy the whole disk, select option **2**. After you have made your selection, quit out of **fdisk** menu by entering **q**. If any other operating systems were previously installed on your system, you also see the following warning message:

```
Warning! All data on your disk will be lost!
Do you wish to continue? (y/n)
```

Enter **y** and press **Return** only if you want your UNIX system to occupy the whole disk. This ensures that **fdisk** partitions the whole disk.

NOTE: If you choose option 3, which allocates the remainder of the hard disk for the UNIX system, you must next activate the UNIX partition by selecting option 5. If you do not activate the UNIX partition, your first partition is activated.

Most computers have diagnostic programs that write to the last cylinder of the hard disk. This means that the last cylinder should not be allocated to a partition. The last cylinder is not allocated when you choose option 2 from the **fdisk** menu. If you choose option 3, you should not allocate the last cylinder of the hard disk to the UNIX partition.

17. Press **Return**, and you see the main **fdisk** menu. You have now set up the partition(s) on your hard disk. To continue with the next step in the installation procedure, enter **q** and press **Return**.

    If you have an ST506 or OMTI controller, continue with Step 18.

    If you have a SCSI controller, skip to Step 27.

    NOTE: The SCSI installation does not run **badtrk** (Steps 18-25).

18. Now you see a menu from the program **badtrk**(ADM). With the **badtrk** program, you can scan your hard disk for defective tracks. The program maps any flawed locations to good tracks elsewhere on the disk. It also creates a bad track table, which is a list of all the bad tracks on your hard disk.

    The main **badtrk** menu looks like this:

```
1. Print Current Bad Track Table
2. Scan Disk (You may choose Read-Only or Destructive later)
3. Add Entries to Current Bad Track Table by Cylinder/Head Number
4. Add Entries to Current Bad Track Table by Sector Number
5. Delete Entries Individually from Current Bad Track Table
6. Delete All Entries from Bad Track Table

Please enter your choice or 'q' to quit:
```

    Enter **2**, then press **Return**.

19. You see the following submenu:

```
1. Scan entire UNIX partition
2. Scan a specified range of tracks
3. Scan a specified filesystem

Please enter your choice or 'q' to quit:
```

Select option **1.**

20. After you select the area you want scanned, you are given the choice:

```
1. Quick scan (approximately 7 megabytes/min)
2. Thorough scan (approximately 1 megabyte/min)

Please enter your choice or 'q' to quit:
```

Select option **2.**

21. You are prompted:

```
Do you want this to be a destructive scan? (y/n)
```

Enter **y**. You are warned:

```
This will destroy the present contents of the region you are scanning.
Do you wish to continue? (y/n)
```

Enter **y** and press **Return**. You see the following message:

```
Scanning in progress, press 'q' to interrupt at any time.
```

22. After you respond to the above prompts, the program scans the active partition of the new disk for flaws. The larger your disk, the longer the scanning process takes, so a very large disk may take a while.

As **badtrk** scans the disk, it displays the number of each track it examines, and the percentage of the disk already scanned. Pressing the **q** key at any time interrupts the scan. If you press **q** to interrupt the scan, you do not need to press **Return**. You are then prompted to continue scanning or to return to the main menu.

Whenever **badtrk** finds a defective track, it lists the location of that track using both the sector number and cylinder/head conventions. Defective track information is entered into the table and displayed on the screen. Here is an example of a bad track:

```
WARNING : wd: on fixed disk ctlr=0 dev=0/47 block=31434 cmd=00000020
    status=00005180, sector = 62899, cylinder/head = 483/4
```

23. When the scan is complete, the menu reappears. Select option **1** to see the results of the scan. Your bad track table looks something like this:

```
Defective Tracks
```

	Cylinder	Head	Sector Number(s)
1.	190	3	12971-12987

```
Press <RETURN> to continue
```

Press **Return** to return to the main menu.

NOTE: If there is a flaw in the first few tracks of the UNIX partition, you are returned to the **fdisk** utility (see the previous installation step). Repartition the disk with **fdisk** so that the UNIX partition no longer includes the defective tracks. You have to experiment to determine how many tracks to exclude. Leave these defective tracks unassigned to any operating system. When you leave **fdisk**, **badtrk** is run again and you should scan the disk for further flaws.

This process continues until **badtrk** finds no flaws in the first few tracks.

24. **If your disk comes with a flaw map**, you should enter any flaws from it into the bad track table.

    Because most disk flaws are marginal or intermittent, your disk's flaw map probably lists more bad tracks than the scanning process reveals. If so, you should now add these defective tracks to the bad track table.

    Select either option **3** or option **4** depending upon the format of the flaw map furnished with your disk. Enter the defective tracks, one per line. If you make a mistake, enter **q** and press **Return**. When you see the main **badtrk** menu, select option 5 to delete a track.

25. **If your disk is not furnished with a flaw map**, or you are finished making changes to the bad track table, enter **q** and press **Return**.

26. You are next prompted for the number of tracks to allocate as replacements for those tracks that are flawed. You should allocate at least as many as the recommended number. Enter the number or just press **Return** to use the recommended number that is displayed:

```
Enter the number of bad tracks to allocate space for
(or press <RETURN> to use the recommended value of n):
```

    If you press **Return** and do not enter an alternate value, **badtrk** allocates the recommended number of tracks as replacements. This number is based on the number of bad tracks currently in the table, plus an allowance for tracks that may go bad in the future. If you ever exceed the number of allocated bad tracks, you must reinstall the system. Next, you see a prompt from **divvy**(ADM). The **divvy** program divides a partition into filesystems. You can create up to seven divisions on a single partition, and name them anything you like.

    NOTE: Try to limit your filesystems to 60-80 megabytes or less. System maintenance tools work faster and more efficiently on this size filesystem.

27. You see the main **divvy** menu and a display that shows how your disk is divided similar to the one below:

```
+----------+------------+--------+---+-------------+------------+
| Name     | Type       | New FS | # | First Block | Last Block |
+----------+------------+--------+---+-------------+------------+
|          | NOT USED   |  no    | 0 |           0|       39011|
|          | NOT USED   |  no    | 1 |       39012|       41511|
|          | NOT USED   |  no    | 2 |          -|          -|
|          | NOT USED   |  no    | 3 |          -|          -|
|          | NOT USED   |  no    | 4 |          -|          -|
|          | NOT USED   |  no    | 5 |          -|          -|
|          | NOT USED   |  no    | 6 |       41512|       41521|
| hd1a     | WHOLE DISK |  no    | 7 |           0|       41980|
+----------+------------+--------+---+-------------+------------+

41522 blocks for divisions, 459 blocks reserved for the system

 n[ame]       Name or rename a division.
 c[reate]     Create a new file system on this division.
 t[ype]       Select or change filesystem type on new filesystems.
 p[revent]    Prevent a new file system from being created on this
 s[tart]      Start a division on a different block.
 e[nd]        End a division on a different block.
 r[estore]    Restore the original division table.

Please enter your choice or 'q' to quit:
```

Each row in the **divvy** table corresponds to a filesystem. When you first see the table, there might be one or more filesystems created. You can change the default size of these filesystems with the **start** and **end** commands. Note that filesystem boundaries must not overlap. For example, filesystem 0 cannot end on the block number where filesystem 1 begins.

When you first see the main **divvy** menu, the filesystems are not named. Use the **name** command to change the name of a filesystem. Filesystems can have any name you choose. For example, you could name a filesystem *u* (for "user").

Do not change the configuration of filesystem 7; it is reserved for internal use by the operating system.

Once you have defined the start and end points of your filesystems, be sure to use the **create** command for each filesystem to make the filesystems on the disk.

The default filesystem type is AFS. If you wish to create filesystems of other types, you must use the **type** command.

Exit from **divvy** by entering **q**. The program prompts whether to install the new partition table, return to the main menu or exit the program without installing the partition table. Select option **i** to install the partition table.

For more information, see the **divvy**(ADM) manual page.

28. The system now creates the filesystems and swap area on your hard disk. This takes several minutes. You see the following message:

```
Making filesystems
```


Adding the New Filesystem

Before leaving system maintenance mode, you must add the new filesystem to the system. To do this, follow these steps:

1. Enter the following command:

   mkdev fs

   Δ **sysadmsh** users select: Filesystems→Add

2. You see the following: Filesystem Initialization Program

```
This program performs maintenance tasks required to add or delete
an existing filesystem. Would you like to:
        1. Add a new filesystem to system.
        2. Remove a filesystem.
Select an option or enter q to quit:
```

   Select **1**.

3. You are next prompted for the device name:

```
Enter a device name and press <Return> or q to quit:
```

Enter the full pathname of the device from */dev/*. For example, to add a filesystem called *u*, you enter */dev/u*.

4. You are now prompted to provide the name of the mount point to be used:

```
Enter a directory name and press <Return> or q to quit:
```

This directory is where the filesystem will be mounted. For example, a filesystem called *u* is mounted on the directory */u*.

5. The following is displayed:

```
Reserving slots in lost+found directory ...

When entering multiuser mode:
        1. Always mount filesystem
        2. Never mount filesystem
        3. Prompt before mounting filesystem

Select an option:
```

If you want the filesystem mounted automatically at system startup, enter **1**. If you wish it mounted only by the request of the system administrator, select **2**. If you select **3**, the system will prompt you at system startup whether or not you want the filesystem mounted.

6. You are then asked whether or not you want to permit users to mount filesystems:

```
Do you want to allow users to mount this file system? (y/n)
```

You must respond **y** so that the system backup program can mount and unmount the filesystem as necessary.

7. The following messages are displayed when the process is complete.

```
Updating system files ...
Filesystem has been successfully added.
```

8. Next, you should mount the */x* filesystem using the following command:

   mount /dev/x /x

   Δ **sysadmsh** users select: Filesystems→Mount

9. To ensure that the system properly recognizes the new filesystem, enter the following commands:

   chmod 755 /dev/x
   chgrp auth /dev/x

   The new filesystem is ready for use.

Relinking the Kernel

If you responded "no" when prompted to relink the kernel after installing a SCSI disk, you must run **link_unix** to manually rebuild the kernel with the new configuration information. Enter the following commands:

cd /etc/conf/cf.d
./link_unix

Δ **sysadmsh** users select: System→Configure→Kernel→Rebuild

Administering ODT-NET

ODT-NET is based on technology developed for TCP/IP and NFS by Lachman Associates, Inc.

12/21/89-1.0.0E
Processed: Wed Dec 20 11:01:48 PST 1989

Contents

Chapter 1

Overview

After you have installed your system following the procedures described in the *Installation Guide*, this guide provides background and reference material for you as the network system administrator. You might need to refer to this information to understand how the network software operates so you can customize your network configuration.

The first part of this chapter presents networking concepts and introduces the four networking products that make up ODT-NET: TCP/IP, SCO™ NFS™, SCO LAN Manager, and UUCP. The latter part of this chapter provides a list of common network administration tasks and directs you to the sections within this guide that help you perform the tasks.

Chapters 2, 3, and 4 relate to TCP/IP, which is the underlying software layer of the network. All network system administrators should become familiar with the general information about TCP/IP presented in Chapter 2. You need to read Chapters 3 and 4 only if you plan to use the optional network services (name server and clock synchronization).

Chapters 5, 6, and 7 contain detailed administrative information about SCO NFS, SCO LAN Manager, and UUCP.

Within the following chapters, you are often referred to the manual pages for further information. You will find all the networking manual pages referred to in this guide together in the "Networking Commands" section of **xman**(X).

Networking Concepts

A distributed network of machines, such as the sample network in Figure 1-1, can provide more aggregate computing power than a mainframe computer, with far less variation in response time over the course of the day. Thus, a network is generally more cost-effective than a central mainframe.

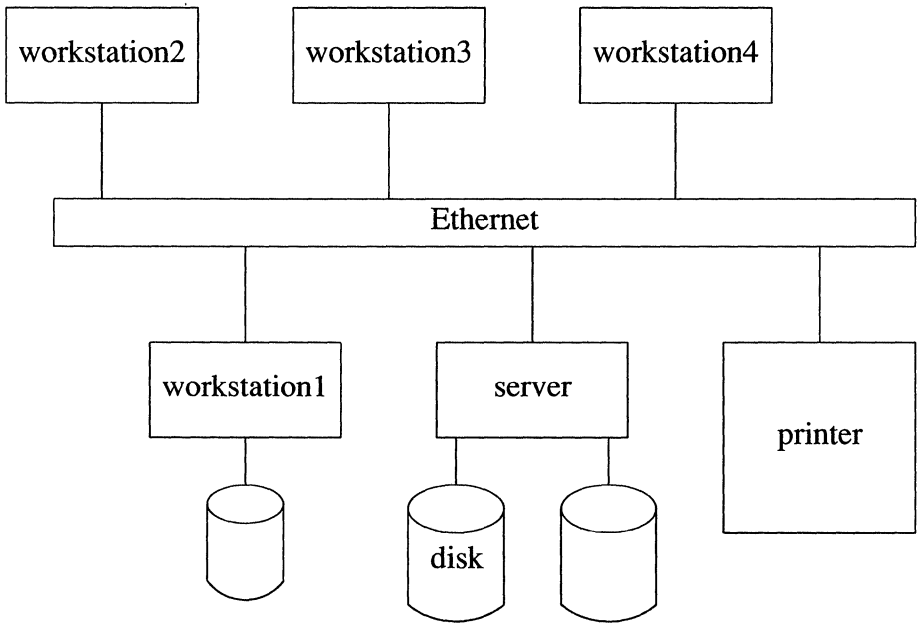


Figure 1-1. Sample Network

The network system administrator oversees the maintenance and operation of the network and is responsible for:

- installing and maintaining network hardware
- installing the network software on each network computer
- assigning names to each computer and device on the network
- assigning names to network users and groups
- adding new users, groups, hardware and software to the network
- performing the commands required to share, remove, and restrict resources
- checking that the required special files are correct

Network Names

Each network computer, user, and group must be identified by a unique name. The network programs use this name to locate computers and to validate network requests.

Network names must be fifteen characters long or less. The first fifteen characters in a network name must be unique to the computer, user, or group using the name. Names cannot be duplicated on the network. For example, you could have computers named "onecomputername" and "twocomputername," but not "computernamesix" and "computernamesixteen." In the last pair of computer names, the first fifteen letters are identical. Each computer name must also be unique regardless of case. For example, the network cannot support a machine named "Bob" and a machine named "bob." For this reason, computer names should use only lowercase letters.

Consistency of Network ID Files

It is imperative that the administration of the computers on the network be consistent. Many functions of the network depend upon a consistent user identification number (user ID) for each individual user on all network machines. If you are connecting existing computers in a network for the first time, be certain that user IDs are consistent from computer to computer. User IDs are defined by the **sysadmsh**(ADM) **Accounts→User→Create** selection and placed in the UNIX® system database files (including */etc/passwd*).

For information on adding user accounts, see the "Administering User Accounts" chapter in *Administering ODT-OS* in the *Administrator's Guide*. For information on maintaining ID consistency, see the **passwd**(C), and *passwd*(F) manual pages.

If you are adding a new computer to the network, or building a network using computers that have never run the UNIX system, make sure the user and group IDs in the new machine's */etc/passwd* and */etc/group* files do not conflict with any other machines on the network.

NOTE: Never edit */etc/passwd* with a text editor. This could damage the password database information and cause your system to refuse further logins. Always use the **sysadmsh**(ADM) **Accounts→User→Create** selection to administer accounts.

User ID numbers must not be duplicated among different users. An individual user can have many different account names, but all account names belonging to an individual user must have the same user ID number.

For example, the user Joe Sparks has accounts on different network computers under the names "joesp", "sparks", and "jsparks". All three of these accounts must have the same user ID number. If the account "joesp" is associated with the user ID number 301, then the accounts under the name "sparks" and "jsparks" must also have user ID number 301.

NOTE: If user ID numbers are not compatible on all network machines, the results of some network commands can be misleading. For example, if a user has different ID numbers on "machine1" and "machine2," and this user tries to access a directory on "machine2" as a user on "machine1," then the local user ID number on "machine1" determines the file ownership information. This means that files on "machine2" may not be accessible because the individual is not recognized across the network.

The entries for accounts belonging to Joe Sparks look like this:

in //machine1/etc/passwd:

```
joesp::301:110:Joe Sparks:/usr/joesp:/bin/csh
```

in //machine2/etc/passwd:

```
sparks::301:110:Joe Sparks:/usr/sparks:/bin/csh
```

in //machine3/etc/passwd:

```
jsparks::301:110:Joe Sparks:/u/jsparks:/bin/sh
```

TCP/IP Background

TCP/IP is a set of protocols used to interconnect computer networks and to route traffic among many different computers. "TCP" means Transmission Control Protocol, and "IP" means Internet Protocol. Protocols are standards that describe allowable formats, error handling, message passing, and communication standards. Computer systems that conform to communications protocols such as TCP/IP are thus able to speak a common language. This enables them to transmit messages accurately to the correct destination, despite differences in the hardware and software of the various machines.

Many large networks conform to these protocols, including the DARPA Internet (Defense Advanced Research Projects Agency Internet). A variety of universities, government agencies, and computer firms are connected to an internetwork that follows the TCP/IP protocols. Thousands of machines are connected to this internet. Any machine on the internet can communicate with any other. (The term internetworking refers to the action of joining two or more networks together. The result can be described as a network of networks, which is called an "internet.") Machines on the internet are referred to as "hosts" or "nodes."

TCP/IP provides the basis for many useful services, including electronic mail, file transfer, and remote login. Electronic mail is designed to transfer short text files. The file transfer application programs transfer very large files containing programs or data. They also provide security checks controlling file transfer. Remote login allows users on one computer to log in at a remote machine and carry on an interactive session.

The Internet Protocol (IP)

The Internet Protocol, IP, defines a connectionless packet delivery. This packet delivery connects one or more packet-handling networks into an internet. The term "connectionless" means that the sending and receiving machines are not connected by a direct circuit. Instead, individual packets of data (datagrams) are routed through different machines on the internet to the destination network and receiving machine. Thus, a message is broken up into several datagrams that are sent separately. Note that connectionless packet delivery by itself is not reliable. Individual datagrams may or may not arrive, and they probably will not arrive in the order in which they were sent. TCP adds reliability.

A datagram consists of header information and a data area. The header information is used to route and process the datagram. Datagrams may be fragmented into smaller pieces, depending on the physical requirements of the networks they cross. (When a gateway sends a datagram to a network that cannot accommodate the datagram as a single packet, the datagram must be fragmented into pieces that are small enough for transmission.) The datagram fragment headers contain the information necessary to reassemble the fragments into the complete datagram. Fragments do not necessarily arrive in order; the software module implementing the IP protocol on the destination machine must reassemble the fragments into the original datagram. If any fragments are lost, the entire datagram is discarded.

The Transmission Control Protocol (TCP)

The Transmission Control Protocol,TCP, works with IP to provide reliable delivery. It provides a means to ensure that the various datagrams making up a message are reassembled in the correct order at their final destination and that any missing datagrams are sent again until they are correctly received.

The primary purpose of TCP is to provide a reliable, secure, virtual-circuit connection service between pairs of communicating processes on top of unreliable subnetworking of packets, where loss, damage, duplication, delay, or misordering of packets can occur. Also, security provisions such as limiting user access to certain machines can be implemented through TCP.

TCP is concerned only with total end-to-end reliability. It makes few assumptions about the possibility of obtaining reliable datagram service. If a datagram is sent across an internet to a remote host, the intervening networks do not guarantee delivery. Likewise, the sender of the datagram has no way of knowing the routing path used to send the datagram. Source-to-destination reliability is provided by TCP in the face of unreliable media; this makes TCP well-suited to a wide variety of multi-machine communication applications.

Reliability is achieved through checksums (error detection codes), sequence numbers in the TCP header, positive acknowledgment of data received, and retransmission of unacknowledged data.

Protocol Layering

Communications software protocols are divided into different layers, where the lowest layer is the hardware that physically transports the data, and the highest layer is the applications program on the host machine. Each layer is very complex in its own right, and no single protocol could encompass all the tasks of the various layers. As discussed earlier, the Internet Protocol handles the routing of datagrams, while the Transmission Control Protocol, which is the layer above IP, provides reliable transmission of messages that were divided into datagrams. The applications programs in turn rely on TCP to send information to the destination host.

TCP/IP has four software layers built on an underlying hardware layer. Its model is shown in Figure 1-2.

Layer	Functionality
4	Application
3	Transport
2	Internet
1	Network Interface

Figure 1-2. TCP/IP Model

The layers operate as follows:

Network Interface Layer	Accepts and transmits data over the network
Internet Layer	Takes care of communication among machines
Transport Layer	Provides communication between application programs
Application Layer	Accesses the Internet, and sends and receives data

To the applications programs, TCP/IP appears to provide a full-duplex virtual circuit between the machines. In actuality, all information is divided into datagrams, which may then be further fragmented during transmission. The software modules implementing IP then reassemble the individual datagrams, while the modules implementing TCP make sure that the various datagrams are reassembled in the order in which they were originally sent.

There are several higher-level specialized protocols for specific applications such as terminal traffic (**telnet**(TC)) and file transfer (**ftp**(TC)), and protocols for other network functions such as gateway-status monitoring. In this manual, however, these are not usually referred to as protocols, but rather as programs or services.

Message Routing

The following sections explain gateways and network addresses. These two concepts are the key to understanding how datagrams are routed through an internet.

Gateways

The various networks that compose an internet are connected through gateway machines. A gateway is a machine that is connected to two or more networks. It can route datagrams from one network to another. Gateways route the datagrams based on the destination network, rather than the individual machine (host) on that network. This simplifies the routing algorithms. The gateway decides which network should be the next destination of a given datagram. If the destination host for the datagram is on that network, the datagram can be sent directly to that host. Otherwise, it continues to pass from gateway to gateway until it reaches the destination network.

Network Addresses

Each host machine on a TCP/IP internet has a 32-bit network address. The address includes two separate parts: the network ID and the host machine ID. Machines that serve as gateways thus have more than one address, because they are on more than one network. Internet addresses are assigned by the Network Information Center (NIC) located at SRI International in Menlo Park, California. The NIC assigns only network IDs; the individual network administrators then assign the host machine IDs for their network.

There are three classes of network addresses, corresponding to small, medium, and large networks. The larger the network, the larger the number of hosts on that network; likewise, smaller networks have fewer hosts. Thus, when the 32-bit network address is divided between the network ID and the host machine ID, larger networks need a larger number of bits to specify all the hosts on the network uniquely. Also, there are only a small number of really large networks, and so fewer bits are needed to identify these networks uniquely. The

network addresses are thus divided into three classes, identified as A, B, or C. The following table lists these classes and their formats.

Class	Network Size Configuration
Class A	Allocates a 7-bit network ID and a 24-bit host ID
Class B	Allocates a 14-bit network ID and a 16-bit host ID
Class C	Allocates a 21-bit network ID and an 8-bit host ID

All network addresses are 32 bits. The first bit of a Class A address is **0** (zero), identifying the address as Class A. Class B addresses begin with the digits **10**, and Class C addresses begin with **11**.

This system of network address classes provides a unique address for the entire statistical distribution of types of networks that might be expected among the various networks using this address system. There are a smaller number of large networks, having many hosts (Class A), a larger number of small networks, consisting of a lesser number of hosts (Class C), and a medium number of networks made up of a medium number of hosts (Class B).

Network addresses are often written as four decimal integers separated by periods (.), where each decimal number represents one octet of the 32-bit network address. For example, a machine might have the address 128.12.3.5.

How NFS Fits into the Network

Even in a network environment, sharing programs and data can sometimes be difficult. Either files have to be copied to each machine where they are needed, or users have to log into the remote machine with the required files. Network logins are time-consuming, and multiple copies of a file become confusing as incompatible changes are made to different copies.

To solve this problem, a distributed filesystem was designed to permit client systems to access shared files on a remote system. Client machines, also called clients, request resources provided by other machines, which are called servers. A server machine makes particular filesystems available, and client machines can mount these as local filesystems. Users can then access remote files as if they were on the local machine.

SCO Network File System (NFS) is a software product that allows you to mount directories across the network and then treat remote files as if they were local. NFS was developed as a standard for the exchange of data between different machines and operating systems.

NFS fits into the application layer of the TCP/IP model. NFS was designed to fit into the network services architecture and not to extend the operating system onto the network. Thus, NFS is an interface to allow a variety of machines and operating systems to play the roles of client and server, but it is not a distributed operating system.

The goal with NFS is to make all disks available as needed. Individual workstations have access to all information residing anywhere on the network. Printers and supercomputers may also be available on the network.

LAN Manager Connects to Microsoft LAN Manager

LAN Manager provides a distributed filesystem with a Microsoft® LAN Manager network. In this local-area network, OS/2®, DOS, and UNIX system machines use TCP/IP to share files.

LAN Manager is compatible with both IBM® PC-Network, Microsoft MS-NET, and Microsoft LAN Manager software, allowing several systems to share files across a local-area network. LAN Manager provides an easy-to-use, flexible, and powerful means of merging multiple OS/2, DOS, and UNIX system machines into a single network.

A LAN Manager network is made up of one or more server computers that control network resources, and one or more client (or consumer) computers that use the resources of the servers. Each computer is connected to the network by a cable. Requests are sent by the consumers and acted upon by a server, then returned to the consumer.

There are certain protocols necessary for the computers to verify requests and perform actions. Controlling these protocols is the function of LAN Manager. LAN Manager uses the "session" or "transport layer" interface provided by the network transport hardware. This interface makes LAN Manager independent of individual network hardware configurations.

When a user on a consumer machine uses a LAN Manager command to access a server computer, the request is passed through various software and hardware layers. These include the LAN Manager distributed filesystem protocol, NetBIOS software and hardware, a transport subsystem, and the cable connecting the machines.

Your Open Desktop™ system can act only as a consumer (or client) to access files from a DOS or OS/2 server; it cannot act as a server to the DOS or OS/2 machine. Through LAN Manager, your Open Desktop system can also consume from a XENIX-NET server. For information about providing access to a XENIX-NET server, see the documentation provided with the XENIX-NET server.

Wide-Area Networks through UUCP

To communicate with computers outside of the local TCP/IP network, the **uucp**, **cu**, and **ct** programs provide access to wide-area networks through telephone lines.

UUCP is a series of programs that provide file-transfer and remote execution capabilities. **cu** and **ct** provide interactive terminal sessions with a computer at a remote site.

cu connects your computer to a remote computer so you can be logged in on both at the same time. You can transfer files or execute commands on either computer without dropping the initial link.

ct connects your computer to a remote terminal so the user of the remote terminal can log in. The user of a remote terminal can call the computer and request that the computer call it back. The computer then drops the initial link so that the remote terminal's modem is available when it is called back.

Common Network Administration Tasks

Certain common tasks are performed by many network system administrators. The purpose of this section is to quickly describe those tasks, then direct you to where you can find the detailed information you need to complete the task. The tasks covered are:

- adding to the hosts database
- setting up routing tables
- establishing user equivalence
- setting up anonymous **ftp**
- mounting a remote filesystem at boot time
- using the name server

Adding to the Hosts Database

The */etc/hosts* file is a list of hosts on the network. Network library routines and server programs use this file to translate between host names and DARPA Internet addresses when the name server is not being used. (Chapter 5 describes how to use the name server.)

To add a machine to your network, you can add an entry to the */etc/hosts* file. Refer to the *hosts*(SFF) manual page for a description of the file format.

Setting Up Routing Tables

Routing tables provide the information needed to properly route packets to their destinations. See "Routing" in Chapter 2 for descriptions of two possible approaches for maintaining routing information. "Network Tuning and Troubleshooting" contains a section on obtaining information about the system routing tables.

Establishing User Equivalence

You can control who has access to a machine through the network by establishing user equivalence within the */etc/hosts.equiv* file. The **rlogin**, **rcp**, and **rcmd** commands use this file to verify access privileges. "Network Databases" in Chapter 2 contains a section that explains how to use this file. Refer also to the *hosts.equiv*(SFF) manual page for a description of the file format.

Setting Up Anonymous ftp

You can set up a public **ftp** account on your system for remote users to transfer files. You can restrict access to certain protected directories within the **ftp** home directory. "Network Databases" in Chapter 2 contains a section about the */etc/ftpusers* file, which also describes how to set up the public **ftp** account.

Mounting a Remote Filesystem at Boot Time

You can set up your distributed filesystem so that a remote filesystem is mounted when your system boots. To do this, you can add an entry to the */etc/default/filesys* file. You must also create the directory through which your system will access the remote filesystem and ensure that your system is listed in the */etc/exports* file on the server. "Introducing NFS" and "Setting Up an NFS Client" in Chapter 5 explain in detail how this is done.

Using the Name Server

Instead of using the */etc/hosts* file for host table lookup, you can use the Berkeley Internet Name Domain (BIND) service for storing and retrieving host names and addresses. To set up this optional network service, you should read Chapter 3.

Chapter 2

TCP/IP Network Administration

This chapter covers topics related to setting up and administering your ODT-NET TCP/IP network. When you installed your system, many of these tasks were performed automatically to configure a basic networked system. If you want to customize your installation or expand your network, you should read this chapter.

If your network is not performing well, the section "Network Tuning and Troubleshooting" at the end of this chapter might provide helpful suggestions.

Kernel Configuration

The following table lists the drivers that must be included in the kernel, along with their associated device nodes.

Name	Device Node	Description
arp	/dev/inet/arp	Address Resolution Protocol
arpproc	(none)	
ip	/dev/inet/ip	Internet Protocol
icmp	/dev/inet/icmp	Internet Control Message Protocol
tcp	/dev/inet/tcp	Transmission Control Protocol
udp	/dev/inet/udp	User Datagram Protocol
llcloop	/dev/llcloop	Loopback interface
socket	/dev/socksys	Socket compatibility package
cp	/dev/socksys1	Copy protection driver
vty	/dev/ptyp*nn*	Virtual TTY driver†
ttyp	/dev/ttyp*nn*	

† The Virtual TTY driver is used by **rlogin**(TC) and **telnet**(TC). There must be one ptyp device and one ttyp device for each virtual TTY configured. Following ptyp or ttyp in the device node name is a two-digit hexadecimal number corresponding to the minor number of the device. For example, **vty** minor 0 is referenced by device node */dev/ptyp00*, and **ttyp** minor 0 is referenced by device node */dev/ttyp00*.

In addition to the drivers listed above, you may also include one or more drivers for your network interface hardware:

Name	Device Node	Description
e3A*nn*	/dev/e3A*nn*	3COM 3C501 ethernet board
e3B*nn*	/dev/e3B*nn*	3COM 3C503 ethernet board
wdn*n*	/dev/wdn*n*	Western Digital WD8003E ethernet board
sl*n*	/dev/slip	Serial Line IP interface

The character *n* in the device nodes indicates any one of the digits 0 through 3. That is, up to four boards of each type are supported. If there were two 3COM 3C503 Ethernet boards, their device nodes would be */dev/e3A0* and */dev/e3A1*.

The interrupt vectors you choose for the various Ethernet boards should be consistent with your hardware requirements.

All drivers must have references in the following files:

- An entry in */etc/conf/cf.d/mdevice*
- A file corresponding to that driver in the */etc/conf/sdevice.d* directory
- An entry in */etc/conf/cf.d/sdevice*

These drivers are normally added to the kernel configuration during installation of TCP/IP. The following display shows the information from a partial *mdevice* file:

```
ip       ocis    iSc     ip      0       23      0       256     -1
rip      ocis    iSc     rip     0       24      0       256     -1
socket   ocrwis  ic      sock    0       25      0       256     -1
ttyp     ocrwi   ict     ttyp    0       26      0       16      -1
vty      ocrwi   ic      vty     0       27      0       16      -1
arpproc  oci     iS      app     0       0       0       256     -1
e3A0     I       iScH    e3c     0       28      1       1       -1
icmp     ocis    iSc     icmp    0       29      0       256     -1
llcloop  ocis    iSc     lo_     0       30      0       256     -1
slip     s       iSc     sl      0       31      0       256     -1
tcp      ocis    iSc     tcp     0       33      0       256     -1
udp      ocis    iSc     udp     0       35      0       256     -1
```

Some of the information in this file may vary depending on the system configuration. In these cases, the numbers that are used depend on the specific system configuration and are probably different from the values shown in this example.

Column six contains the major block device number, which varies depending upon the drivers that were installed in the system and the order in which they were installed. The actual value for any given driver does not actually matter as long as each driver has a different number and the number in this file matches the major number of the device name in the */dev* directory that is supposed to refer to it. The **arpproc** module is a special case, as it has no corresponding pathname in */dev*; for this driver, the block major device number is 0.

The following is a partial *sdevice* file (comments have been removed for clarity):

```
sio     Y       1       7       1       4       3f8     3ff     0       0
sio     Y       1       7       1       3       2f8     2ff     0       0
slip    Y       256     0       0       0       0       0       0       0
socket  Y       256     0       0       0       0       0       0       0
sp      Y       0       0       0       0       0       0       0       0
spt     Y       0       0       0       0       0       0       0       0
ss      Y       0       0       0       0       0       0       0       0
str     Y       0       0       0       0       0       0       0       0
svdsp   Y       1       0       0       0       0       0       0       0
svkbd   Y       1       0       0       0       0       0       0       0
sxt     N       1       0       0       0       0       0       0       0
sy      Y       1       0       0       0       0       0       0       0
tcp     Y       256     0       0       0       0       0       0       0
timod   Y       1       0       0       0       0       0       0       0
tirdwr  Y       1       0       0       0       0       0       0       0
```

The *sdevice* file is actually assembled from component files in the directory */etc/conf/sdevice.d*. Each component file contains the line describing that driver. The "Y" or "N" in the second column indicates whether the the driver is to be linked into the kernel. Column six is the interrupt vector, which varies depending upon which cards are in the system and the vectors for which they are setup.

The format of these files is defined in **sdevice**(F) and **mdevice**(F).

Runtime Configuration of STREAMS Drivers

STREAMS configuration (linking the various STREAMS drivers and modules together) is handled by the **slink**(ADMN) program, which is normally executed at boot time by **tcp**(ADMN). The **slink** program reads the file */etc/strcf*, which contains a list of STREAMS operations to perform. Most of */etc/strcf* is the same on every system. However, under unusual circumstances, it may be necessary to edit the section of */etc/strcf* that configures the network interfaces. Examples for various types of network drivers are provided. In some cases, it is necessary to write new driver setup procedures. See **slink**(ADMN) and **strcf**(SFF) for further information.

SLIP drivers are handled automatically by the **slattach**(ADMN) command, which is invoked in the */etc/tcp* script. This portion of the script is set up during installation of the SLIP driver.

The following sections present examples of **slink** configuration commands for several different driver types.

Cloning Drivers with One Major Number per Interface

Drivers of this type, such as the 3COM 3C503 *e3B0* driver or Western Digital WD8003E *wd* driver, use cloning but do not support a method of selecting a particular network interface (such as **unit select**). Rather, this is done by allocating a separate major device number to each network interface. The **slink** function **cenet** configures an interface of this type. The command line to configure such an interface has the form:

```
cenet ip /dev/e3B0 e3B0 0
```

To add a second interface, add the following line:

```
cenet ip /dev/e3B0 e3B0 1
```

Note that the device node actually used is formed by concatenating the given device node name prefix (*/dev/e3B0*) and the given unit number (*0* or *1*). The interface name is formed in a similar manner using the supplied interface name prefix (*e3B0*) and the unit number. Thus, the first example configures an interface named *e3B0*, which accesses the device referred to by */dev/e3B0*.

Cloning Drivers Using unit select or DL_ATTACH

These drivers have only one device node and one major number, which are used for all interfaces. (The SLIP drivers are of this type, but they are a special case in that individual SLIP interfaces do not need explicit configuration in */etc/strcf*. The STREAMS configuration of SLIP drivers is handled by the **slattach**(ADMN) command, which is invoked from */etc/tcp* during system startup. The appropriate **slattach** command is automatically placed in the */etc/tcp* file during installation of TCP/IP Runtime.) The desired interface is selected using either the **unit select** or the **DL_ATTACH** primitive. (Normally, a given driver recognizes only one of these primitives.) A primitive is a type of command used to invoke a primitive operation. A primitive operation can be described as part of an interface between two programs or pieces of software. In this case, a primitive operation is a service provided by one of the protocol layers.

The **slink** functions **uenet** and **denet** configure this type of driver; **uenet** uses **unit select**, while **denet** uses **DL_ATTACH**. The command line to configure an interface of this type has the form:

```
uenet ip /dev/abc en 0
```

For a driver that uses **DL_ATTACH**, use **denet** in place of **uenet**. To configure a second interface, add the following line:

```
uenet ip /dev/abc en 1
```

The **denet** and **uenet** functions form the interface name in the same manner as does **cenet** (see previous section), but the device node name is unchanged (*/dev/abc* is open in both of these examples).

Non-Cloning Drivers

Drivers of this type have a separate device node for each minor device, with some fixed number of minor devices allocated to each network interface. The **slink** functions **senetc** and **senet** are used for this driver type. (The **senetc** function allows the specification of a convergence module.) The following command line configures such an interface:

```
senetc ip eli /dev/emd0 /dev/emd1 en0
```

If a convergence module is not required, use **senet** in place of **senetc** and omit "eli."

The last argument (*en0* in this example) gives the name by which the newly created interface is known for the purpose of performing interface-configuration operations via **ifconfig**(ADMN). For further information, refer to the section entitled "Setting Interface Parameters" later in this chapter.

Assuming that there are four minor devices assigned to each network interface, a second interface would be configured as follows:

```
senetc ip eli /dev/emd4 /dev/emd5 en1
```

Setting Interface Parameters

All network interface drivers, including the loopback interface, require that their host addresses be defined at boot time. This is done with **ifconfig**(ADMN) commands included in the */etc/tcp* shell script. These commands are normally set up automatically during installation. This configuration applies only to simple, basic configurations. For example, if you want to use the network feature of **ifconfig**, you need to edit */etc/tcp* manually and modify the **ifconfig** commands there.

ifconfig can also be used to set options for an interface at boot time. Options are set independently for each interface and apply to all packets sent using that interface. These options include disabling the use of the Address Resolution Protocol. This may be useful if a network is shared with hosts running software that does not yet provide this function. Alternatively, translations for such hosts can be set in advance or published by a UNIX System host by use of the **arp**(ADMN) command.

Local Subnetworks

In TCP/IP, the DARPA Internet support includes the concept of the subnetwork. This is a mechanism that enables several local networks to appear as a single Internet network to off-site hosts. Subnetworks are useful because they allow a site to hide the local topology, requiring only a single route in external gateways. This also means that local network numbers may be locally administered.

To set up local subnetworks, you first need to know how much of the available address space is to be partitioned. The term "address" is used here to mean the Internet host part of the 32-bit address. Sites with a class A network number have a 24-bit address space with which to work, sites with a class B network number have a 16-bit address space; and sites with a class C network number have an 8-bit address space. To define local subnets you must steal some bits from the local host address space for use in extending the network portion of the internet address.

This reinterpretation of internet addresses is done only for local networks. It is not visible to off-site hosts. For example, if your site has a class B network number, hosts on this network have an Internet address that contains the network number, 16 bits, and the host number,

another 16 bits. To define 254 local subnets, each possessing at most 255 hosts, 8 bits may be taken from the local part to be used for the subnetwork ID. (The use of subnets 0 and all-*1*'s, 255 in this example, is discouraged to avoid confusion about broadcast addresses.) New network numbers are then constructed by concatenating the original 16-bit network number with the extra 8 bits containing the local subnetwork number.

The existence of local subnetworks is communicated to the system when a network interface is configured with the **netmask** option to the **ifconfig**(ADMN) program. A network mask defines the portion of the internet address that is to be considered the network part for that network. This mask normally contains the bits corresponding to the standard network part as well as the portion of the local part that was assigned to subnets. If no mask is specified when the address is set, a mask is set according to the class of the network. For example, at Berkeley (class B network 128.32), 8 bits of the local part are reserved for defining subnetworks. Consequently, the */etc/tcp* file contains lines of the form:

```
/etc/ifconfig e3B0 netmask 0xffffff00 128.32.1.7
```

This specifies that for interface *e3B0*, the upper 24 bits of the internet address should be used in calculating network numbers (netmask 0xffffff00). The internet address of the interface is 128.32.1.7 (host 7 on network 128.32.1). Hosts *m* on subnetwork *n* of this network would then have addresses of the form 128.32.*n.m*. For example, host 99 on network 129 would have an address 128.32.129.99. For hosts with multiple interfaces, the network mask should be set for each interface, although in practice only the mask of the first interface on each network is actually used.

Internet Broadcast Addresses

The broadcast address for internet networks is defined according to RFC-919 as the address with a host part of all *1*'s. The address used by 4.2BSD was the address with a host part of *0*. The UNIX System uses the standard broadcast address (all *1*'s) by default, but allows the broadcast address to be set (with **ifconfig**) for each interface. This allows networks consisting of both 4.2BSD and UNIX System hosts to coexist while the upgrade process proceeds. In the presence of subnets, the broadcast address uses the subnet field as for normal host addresses, with the remaining host part set to *1*'s (or *0*'s, on a network that has not yet been converted). The UNIX System hosts recognize and accept packets sent to the logical-network broadcast address as well as those sent to the subnet broadcast address, and, when using an all-*1*'s broadcast, also recognize and receive packets sent to host *0* as a broadcast.

Routing

If your environment allows access to networks not directly attached to your host, you need to set up routing information to allow packets to be properly routed. Two schemes are supported by the system. The first employs the routing table management daemon **routed**(ADMN) to maintain the system routing tables. The routing daemon uses a variant of the Xerox Routing Information Protocol to maintain up-to-date routing tables in a cluster of local-area networks. By using the */etc/gateways* file, the routing daemon can also initialize static routes to distant networks. (See the next section for further discussion.) When the routing daemon is started (usually from */etc/tcp*), it reads */etc/gateways* if it exists and installs those routes defined there. It then broadcasts on each local network to which the host is attached to find other instances of the routing daemon. If any responses are received, the routing daemons cooperate in maintaining a globally consistent view of routing in the local environment. This view can be extended to include remote sites also running the routing daemon by setting up suitable entries in */etc/gateways*. See **route**(ADMN) for a more thorough discussion.

The second approach is to define a default or wildcard route to a smart gateway and depend on the gateway to provide ICMP routing redirect information to create dynamically a routing data base. This is done by adding an entry to */etc/tcp* as in the following example:

```
/etc/route add default smart-gateway 1
```

See **route**(ADMN) for more information. The system uses the default route as a last resort in routing packets to their destinations. Assuming the gateway to which packets are directed can to generate the proper routing redirect messages, the system then adds routing table entries based on the information supplied. This approach has certain advantages over the routing daemon, but it is unsuitable in an environment where there are only bridges. (For example, pseudo-gateways do not generate routing-redirect messages.) Further, if the smart gateway goes down, there is no alternative, save manual alteration of the routing table entry, to maintain service.

The system always listens to, and processes, routing redirect information, and so it is possible to combine both of the above facilities. For example, the routing table management process might be used to maintain up-to-date information about routes to geographically local networks, while employing the wildcard routing techniques for distant networks. The **netstat**(TC) program displays routing table contents as well as various routing-oriented statistics. The following example displays the contents of the routing tables:

```
netstat -r
```

Alternatively, the following shows the number of routing table entries dynamically created as a result of routing redirect messages and so forth:

```
netstat -r -s
```

Using UNIX System Machines as Gateways

Any UNIX System machine that is connected to more than one network functions as a gateway. At a gateway machine, packets received on one network that are destined for a host on another network are automatically forwarded. If a packet cannot be forwarded to the desired destination, an ICMP error message is sent to the originator of the packet. When a packet is forwarded back through the interface on which it arrived, an ICMP redirect message is sent to the source host if it is on the same network. This improves the interaction of UNIX System gateways with hosts that configure their routes via default gateways and redirects.

Local-area routing within a group of interconnected Ethernets and other such networks can be handled by **routed**(ADMN). Gateways between the ARPANET or MILNET and one or more local networks require an additional routing protocol, the Exterior Gateway Protocol (EGP), to inform the core gateways of their presence and to acquire routing information from the core. (EGP is not currently supported in this product.)

Network Servers

In the UNIX System, most of the server programs are started by a super server, called the "internet daemon." The internet daemon, */etc/inetd*, acts as a master server for programs specified in its configuration file, */etc/inetd.conf*, listening for service requests for these servers, and starting up the appropriate program whenever a request is received. The configuration file includes lines containing a service name (as found in */etc/services*), the type of socket the server expects (for example, stream or dgram), the protocol used with the socket (as found in */etc/protocols*), whether to wait for each server to complete before starting up another, the user name under which the server should run, the server program's name, and at most five arguments to pass to the server program. Some trivial services are implemented internally in **inetd**(SFF), and their servers are listed as internal. For example, an entry for the file-transfer protocol server would appear as:

```
ftp    stream    tcp    nowait    root    /etc/ftpd    ftpd
```

Consult **inetd**(ADMN) for more details on the format of the configuration file and the operation of the Internet daemon.

Network Databases

Several data files are used by the network library routines and server programs. Most of these files are host independent and updated only rarely. The following table lists the data files used.

File	Manual Reference	Use
/etc/hosts	hosts (SFF)	host names
/etc/networks	networks (SFF)	network names
/etc/services	services (SFF)	list of known services
/etc/protocols	protocols (SFF)	protocol names
/etc/hosts.equiv	rshd (ADMN)	list of "trusted" hosts
/etc/ftpusers	ftpd (ADMN)	list of "unwelcome" **ftp** users
/etc/inetd.conf	inetd (ADMN)	list of servers started by **inetd**

The files distributed are set up for ARPANET or other internet hosts. Local networks and hosts should be added to describe the local configuration. Network numbers must be chosen for each Ethernet. For sites not connected to the Internet, these can be chosen more or less arbitrarily; otherwise, the normal channels should be used for allocation of network numbers.

The /etc/hosts.equiv File

There are several files that are used to establish user equivalence. One is the */etc/hosts.equiv* file, which covers the system as a whole, except for the root account. The other is the *.rhosts* file in the individual user account's home directory. This file covers only the individual user account. (For root, this is */.rhosts*.) These two files work together with a third file, */etc/passwd*, to determine the extent of user equivalence.

There are two ways to establish user equivalence:

- An entry in *.rhosts* and in */etc/passwd*
- An entry in */etc/hosts.equiv* and in */etc/passwd*

In both cases, */etc/passwd* must contain an entry for the user name from the remote machine. However, the two methods have differing scopes. If the file *.rhosts* is used in a particular account, then user equivalence is established for that account only. However, if there is an entry in */etc/hosts.equiv* for a host name and an account on that host, then that account has user equivalence for any account (except root). If the entry in */etc/hosts.equiv* has only the remote host name, then any user on that host has user equivalence for all local accounts (except root). Such a host is considered a "trusted host."

NOTE: Entries in */etc/hosts.equiv* can create large holes in system security. Be sparing in their use. In most circumstances, it is unwise to create entries that allow all users on remote machines to access all accounts on your local machine.

For example, suppose you have an account under the user name "Test1" on machine "Admin." You want to establish user equivalence on the remote machine "Systemb." The administrator for the machine Systemb must add an entry to the */etc/passwd* file for an account name Test1. They must also include the following entry in the file */etc/hosts.equiv* on Systemb:

```
Admin   Test1
```

This gives user equivalence for all accounts except root to user Test1 on the machine Systemb. Suppose that Test1 really only needed access to the account Testb on Systemb. Then it would be better to remove the above entry from */etc/hosts.equiv* on Systemb and use the following entry in the file *.rhosts* in the home directory for Testb:

```
Admin   Test1
```

Note that entries for *.rhosts* must include both the system name and the account name. The file */etc/hosts.equiv* does allow entries for the system name only, as discussed earlier.

If there are entries in both *.rhosts* and */etc/hosts.equiv* for the same machine or machine/account combination, then the entry from */etc/hosts.equiv* determines the extent of user equivalence.

The /etc/ftpusers File

The **ftp** server included in the system provides support for an anonymous **ftp** account. Because of the inherent security problems with such a facility, you should read this section carefully if you want to provide such a service.

An anonymous account is enabled by creating a user called **ftp**. When a client uses the anonymous account, a **chroot**(ADM) system call is performed by the server to restrict the client from moving outside that part of the filesystem where the **ftp** home directory is located. Because a **chroot** call is used, certain programs and files used by the server process

must be placed in the **ftp** home directory. Further, you must be sure that all directories and executable images are unwritable. The following directory setup is recommended:

```
# cd ~ftp
# chmod 555 .; chown ftp .; chgrp ftp .
# mkdir bin etc pub lib dev
# chown root bin etc lib dev
# chmod 555 bin etc lib dev
# chown ftp pub
# chmod 777 pub
# cd bin
# cp /bin/sh /bin/ls .
# chmod 111 sh ls
# cd ../etc
# cp /etc/passwd /etc/group .
# chmod 444 passwd group
# cd ../lib
# cp /shlib/libc_s .
# cd ..
# find /dev/socksys -print | cpio -dumpv .
```

When local users want to place files in the anonymous area, they must place them in a subdirectory. In the setup here, the directory *~ftp/pub* is used.

Another issue to consider is the */etc/passwd* file placed here. It can be copied by users who use the anonymous account. They can then try to break the passwords of users on your machine for further access. A good choice of users to include in this copy might be root, daemon, uucp, and the **ftp** user. All passwords here should probably be *.

Aside from the problems of directory modes and such, the **ftp** server provides a loophole for interlopers if certain user accounts are allowed. The file */etc/ftpusers* is checked on each connection. If the requested user name is located in the file, the request for service is denied. It is suggested that this file contain at least the following names:

```
uucp
root
```

Accounts with nonstandard shells should be listed in this file. Accounts without passwords need not be listed in this file; the **ftp** server does not service these users.

Network Tuning and Troubleshooting

It is likely that from time to time you will encounter problems using your network. The first thing to do is check your network connections. On networks such as the Ethernet a loose cable tap or poorly placed power cable can result in severely deteriorated service. The **ping**(ADMN) command is particularly useful for confirming the existence of network connections. If there is no hardware problem, check next for routing problems and addressing problems.

The **netstat**(TC) program can also be helpful in tracking down hardware malfunctions. In particular, look at the **-i** and **-s** options in the manual page. The **netstat**(TC) program also shows detailed information about network behavior. Examples of **netstat** displays appear later in this chapter.

If you think a communication protocol problem exists, consult the protocol specifications and attempt to isolate the problem in a packet trace. The **SO_DEBUG** option can be supplied before establishing a connection on a socket, in which case the system traces all traffic and internal actions (such as timers expiring) in a circular trace buffer. This buffer can then be printed out with the **trpt**(ADMN) program. Most of the servers distributed with the system accept a **-d** option forcing all sockets to be created with debugging turned on. Consult the appropriate manual pages for more information.

STREAMS Tuning

The **crash**(ADM) command can be used to display STREAMS usage of buffers of various sizes. Typical symptoms of inadequate STREAMS buffer space include the following: lost connections for no reason; processes that communicate over the network hang; and programs that communicate over the network suddenly malfunction. Use the UNIX Link Kit **configure** command to increase STREAMS buffer resources.

Active Connections Display

The active connections display is the default display of the **netstat**(TC) command. It displays a line of information for each active connection on the local machine under the headings described below.

netstat -a

Active Internet connections (including servers) are as follows:

```
scobox$ netstat -a
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address       Foreign Address  (state)
ip         0      0  *.*                 *.*
tcp        0      0  scobox.telnet       scoter.2460      ESTABLISHED
tcp        0      0  *.smtp              *.*              LISTEN
tcp        0      0  *.1024              *.*              LISTEN
tcp        0      0  *.sunrpc            *.*              LISTEN
tcp        0      0  *.chargen           *.*              LISTEN
tcp        0      0  *.daytime           *.*              LISTEN
tcp        0      0  *.time              *.*              LISTEN
tcp        0      0  *.domain            *.*              LISTEN
tcp        0      0  *.finger            *.*              LISTEN
tcp        0      0  *.exec              *.*              LISTEN
tcp        0      0  *.ftp               *.*              LISTEN
tcp        0      0  *.telnet            *.*              LISTEN
tcp        0      0  *.login             *.*              LISTEN
tcp        0      0  *.shell             *.*              LISTEN
tcp        0      0  scobox.listen       *.*              LISTEN
tcp        0      0  scobox.nterm        *.*              LISTEN
tcp        0      0  *.*                 *.*              CLOSED
udp        0      0  *.1035              *.*
udp        0      0  *.1034              *.*
udp        0      0  *.1033              *.*
udp        0      0  *.1032              *.*
udp        0      0  *.2049              *.*
udp        0      0  *.1028              *.*
udp        0      0  *.sunrpc            *.*
udp        0      0  scobox.domain       *.*
udp        0      0  localhost.domain    *.*
scobox$
```

Descriptions of the Display Headings

- The protocol used in the connection.
- Receive queue. The number of received characters (bytes) of data waiting to be processed.
- Send queue. The number of characters (bytes) of data waiting to be transmitted.
- The port number of the local connection, displayed symbolically. The port numbers are taken from the */etc/services* file.
- The port number of the remote connection, displayed symbolically. The port numbers are taken from the */etc/services* file.
- The current state of the connection. Each protocol has its own set of states. For the protocol-dependent states that can be displayed, see the appropriate protocol specification.

Interfaces

This display describes activities on all the local machine's interfaces to the net, in the form of a table of cumulative statistics. This display is available through **netstat** with the **-i** option.

netstat -i

```
scobox$ netstat -i
Name Mtu  Network     Address      Ipkts   Ierrs Opkts   Oerrs Collis
en0  1500 sco-eng-ne  scobox       No Statistics Available
e3B0 1500 128.174.14  128.174.14.1 0       0     0       0     0
lo0  2048 loopback    localhost    189     0     189     0     0
scobox$
```

Descriptions of the Display Headings

Each interface is described by a line with the following headings:

Name	The name of the network interface. For example, *en0* is the name of the first Ethernet interface board.
Mtu	Maximum transmission unit (in bytes). This is the largest size permitted for any single packet sent through this interface.
Network	The name of the network address of the interface as given in */etc/networks*.
Address	The name of the machine address of the interface in */etc/hosts*.
Ipkts	Input packets. The number of packets received on the interface.
Ierrs	Input errors. The number of errors detected in packets of data received on this interface.
Opkts	Output packets. The number of packets transmitted on the interface.
Oerrs	Output errors. The number of errors detected and corrected in packets of data transmitted on this interface.
Collis	Collisions that occurred on the network.

Routing Tables

The Routing Table display provides information about the usage of each route you have configured. A route consists of a destination host or network and a network interface used to exchange packets. Direct routes are created for each interface attached to the local host.

netstat -r

```
scobox$ netstat -r
Routing tables
Destination     Gateway          Flags    Refcnt Use     Interface
localhost       localhost        UH       4      0       lo0
sco-eng-net     scobox           U        4      537     en0
128.174.14      128.174.14.1     U        0      0       e3B0
128.174         scoffle          UG       0      0       en0
scobox$
```

Descriptions of the Display Headings

The information displayed for each route is as follows.

Destination — The network or machine to which this route allows you to connect.

Gateway — The name of the gateway you configured for this route. If you are directly connected, this is a local address. Otherwise, it is the name of the machine through which packets must be routed.

Flags — The state of the route. Valid states are:

U	up
G	a route to a gateway
N	a route to a network
H	a route to a host

Refcnt — The current number of active connections using the route. Connection-oriented protocols normally hold on to a single route for the duration of the connection, while connectionless protocols obtain a route and then discard it as needed.

Use — The current number of packets sent using this route.

Interface — The name of the physical network interface used to begin the route.

Statistics Display

The Protocol Statistics display provides protocol-specific errors. The errors in the display are grouped under headings for each higher-level protocol in your system. The headings are protocol-specific.

- Internet Protocol (ip)
- Internet Control Message Protocol (icmp)
- Transmission Control Protocol (tcp)
- User Datagram Protocol (udp)

netstat -s

```
ip:
        3209 total packets received
        0 bad header checksums
        0 with size smaller than minimum
        0 with data size < data length
        0 with header length < data size
        0 with data length < header length
        0 fragments received
        0 fragments dropped (dup or out of space)
        0 fragments dropped after timeout
        0 packets forwarded
        0 packets not forwardable
        0 redirects sent
icmp:
        1 call to icmp_error
        0 errors not generated because old message was icmp
        Output histogram:
                destination unreachable: 1
```

(Continued on next page.)

(Continued)

```
        0 messages with bad code fields
        0 messages < minimum length
        0 bad checksums
        0 messages with bad length
        Input histogram:
                destination unreachable: 640
        0 message responses generated
tcp:
        348 packets sent
                202 data packets (3661 bytes)
                0 data packets (0 bytes) retransmitted
                101 ack-only packets (60 delayed)
                0 URG only packets
                0 window probe packets
                0 window update packets
                45 control packets
        411 packets received
                233 acks (for 3654 bytes)
                19 duplicate acks
                0 acks for unsent data
                200 packets (1677 bytes) received in-sequence
                0 completely duplicate packets (0 bytes)
                0 packets with some dup. data (0 bytes duped)
                9 out-of-order packets (0 bytes)
                0 packets (0 bytes) of data after window
                0 window probes
                0 window update packets
                0 packets received after close
                0 discarded for bad checksums
                0 discarded for bad header offset fields
                0 discarded because packet too short
        25 connection requests
        12 connection accepts
        21 connections established (including accepts)
        72 connections closed (including 0 drops)
        16 embryonic connections dropped
        233 segments updated rtt (of 259 attempts)
        0 retransmit timeouts
                0 connections dropped by rexmit timeout
```

(Continued on next page.)

(Continued)

```
        0 persist timeouts
        0 keepalive timeouts
                0 keepalive probes sent
                0 connections dropped by keepalive
        0 connections lingered
                0 linger timers expired
                0 linger timers cancelled
                0 linger timers aborted by signal
udp:
        0 incomplete headers
        0 bad data length fields
        0 bad checksums
```

Chapter 3

Name Server Operations Guide for BIND

A name server is a network service that enables clients to name resources or objects and share this information with other objects in the network. The Berkeley Internet Name Domain (BIND) Server implements the DARPA Internet name server for the UNIX operating system. In effect, this is a distributed database system for objects in a computer network. BIND is fully integrated into network programs for use in storing and retrieving host names and addresses. The system administrator can configure the system to use BIND as a replacement for the original host table lookup of information in the network hosts file */etc/hosts*. The default configuration does not use BIND. BIND is initially disabled. If you want to use it, you must first set up the necessary configuration files.

The Name Service

The basic function of the name server is to provide information about network objects by answering queries. The advantage of using a name server over the host table lookup for host-name resolution is to avoid the need for a single centralized clearinghouse for all names. The authority for this information can be delegated to the different organizations on the network responsible for it.

The host table lookup routines require that the master file for the entire network be maintained at a central location by a few people. This works well for small networks where there are only a few machines and the different organizations responsible for them cooperate. However, this does not work well for large networks where machines cross organizational boundaries.

With the name server, the network can be broken into a hierarchy of domains. The name space is organized as a tree, according to organizational or administrative boundaries. Each node, called a domain, is given a label, and the name of the domain is the concatenation of all the labels of the domains from the root to the current domain, listed from right to left, separated by dots. A label need only be unique within its domain. The whole space is partitioned into several areas called zones, each starting at a domain and extending down to the leaf domains or to domains where other zones start. Zones usually represent

administrative boundaries. An example of a host address for a host at the University of California, Berkeley, would look as follows:

```
monet.Berkeley.EDU
```

The top-level domain for educational organizations is EDU; Berkeley is a subdomain of EDU and monet is the name of the host. Additional top-level domains include:

COM	Commercial Organizations
GOV	Government Organizations
MIL	Military Departments
ORG	Miscellaneous Organizations

Types of Servers

There are several types of servers. These are:

- master servers
- caching-only servers
- remote servers
- slave servers

These types of servers are described in more detail in the following four sections.

Master Servers

A master server for a domain is the authority for that domain. This server maintains all the data corresponding to its domain. Each domain should have at least two master servers: a primary master, and some secondary masters to provide backup service if the primary is unavailable or overloaded. A server may be a master for multiple domains, being primary for some domains and secondary for others.

Primary

A primary master server is a server that loads its data from a file on disk. This server may also delegate authority to other servers in its domain.

Secondary

A secondary master server is a server that is delegated authority and receives its data for a domain from a primary master server. At boot time, the secondary server requests all the data for the given zone from the primary master server. This server then periodically checks with the primary server to see if it needs to update its data.

Caching-Only Servers

All servers are caching servers. This means that the server caches the information that it receives for use until the data expires. A caching only server is a server that is not authoritative for any domain. This server services queries and asks other servers that have the authority for the information needed. All servers keep data in their caches until the data expires, based on a time-to-live field attached to the data when it is received from another server.

Remote Servers

A remote server is an option given to people who would like to use a name server on their workstation or on a machine that has a limited amount of memory and CPU cycles. With this option, you can run all of the networking programs that use the name server without running the name server on the local machine. All of the queries are serviced by a name server that is running on another machine on the network.

Slave Server

A slave server is a server that always forwards queries it cannot satisfy locally to a fixed list of forwarding servers instead of interacting with the master name servers for the root and other domains. The queries to the forwarding servers are recursive queries. There may be one or more forwarding servers, and they are tried in turn until the list is exhausted. A slave and forwarder configuration is typically used when you do not wish all the servers at a given site to be interacting with the rest of the Internet servers. A typical scenario would involve a number of workstations and a departmental timesharing machine with Internet access. The workstations might be administratively prohibited from having Internet access. To give the workstations the appearance of access to the Internet domain system, the workstations could

be slave servers to the timesharing machine, which would forward the queries and interact with other name servers to resolve the query before returning the answer. An added benefit of using the forwarding feature is that the central machine develops a much more complete cache of information that all the workstations can take advantage of. The use of slave mode and forwarding is discussed further under the description of the named bootfile commands.

Setting Up Your Own Domain

When setting up a domain that is going to be on a public network, the site administrator should contact the organization in charge of the network and request the appropriate domain registration form. An organization that belongs to multiple networks (such as CSNET, DARPA Internet, and BITNET) should register with only one network.

The contacts are as follows:

DARPA Internet

Sites that are already on the DARPA Internet and need information on setting up a domain should contact HOSTMASTER@SRI-NIC.ARPA. You may also want to be placed on the BIND mailing list, which is a mail group for people on the DARPA Internet running BIND. This group discusses future design decisions, operational problems, and other related topics. To request placement on this mailing list, send mail to the following address:

```
bind-request @ucbarpa.Berkeley.EDU.
```

CSNET

A CSNET member organization that has not registered its domain name should contact the CSNET Coordination and Information Center (CIC) for an application and information about setting up a domain.

An organization that already has a registered domain name should keep the CIC informed about how it would like its mail routed. In general, the CSNET relay prefers to send mail via CSNET if possible (as opposed to BITNET or the Internet). For an organization on multiple networks, this may not always be the preferred behavior. The CIC can be reached via electronic mail at cic @ sh.cs.net, or by phone at (617) 497-2777.

BITNET

If you are on the BITNET and need to set up a domain, contact INFO@BITNIC.

Boot File

The name server uses several files to load its database. The major file used is the boot file. This is the file that is first read when **named** starts up. This tells the server what type of server it is, which zones it has authority over, and where to get its initial data. The default location for this file is */etc/named.boot*. However, this can be changed by setting the BOOTFILE variable when you compile **named** or by specifying the location on the command line when **named** starts up.

Domain

The boot file contains a line of code that designates the default domain. The line for the server looks like this:

```
domain          Berkeley.Edu
```

The name server uses this information when it receives a query for a name without a "." that is unknown. When it receives one of these queries, it appends the name in the second field to the query name. This is an obsolete facility, which will be removed from future releases.

Directory

The directory line specifies the directory in which the name server should run, allowing the other filenames in the boot file to use relative pathnames.

```
directory               /usr/local/lib/named
```

If you have more than a couple of named files to be maintained, you may wish to place the named files in a directory such as */usr/local/domain* and adjust the directory command properly. The main purposes of this command are to make sure **named** is in the proper directory when trying to include files by relative pathnames with $INCLUDE and to allow **named** to run in a location that is reasonable to dump core if it feels the urge.

Primary Master

The line in the boot file that designates the server as a primary server for a zone looks like the following:

```
primary             Berkeley.Edu    ucbhosts
```

The first field specifies that the server is a primary one for the zone stated in the second field. The third field is the name of the file from which the data is read.

Secondary Master

The line for a secondary server is similar to that for the primary, except that it lists addresses of other servers (usually primary servers) from which the zone data is obtained.

```
secondary           Berkeley.Edu    128.32.0.10     128.32.0.4
```

The first field specifies that the server is a secondary master server for the zone stated in the second field. The two network addresses specify the name servers that are primary for the zone. The secondary server gets its data across the network from the listed servers. Each server is tried in the order listed until it successfully receives the data from a listed server. If a filename is present after the list of primary servers, data for the zone is dumped into that file as a backup. When the server is first started, the data are loaded from the backup file if possible, and a primary server is then consulted to check that the zone is still up-to-date.

Caching-Only Server

You do not need a special line to designate that a server is a caching server. A caching-only server is indicated by the absence of authority lines, such as secondary or primary in the boot file.

All servers should have the following line in the boot file to prime the name server's cache:

```
cache               .               root.cache
```

The period (.) specifies the current domain. All cache files listed are read in at named boot time and any values still valid are reinstated in the cache and the root name server information in the cache files are always used. For information on the cache file, see the later section, "Initializing the Cache."

Forwarders

Any server can make use of forwarders. A forwarder is another server capable of processing recursive queries to try to resolve queries on behalf of other systems. The **forwarders** command specifies forwarders by internet address as follows:

```
forwarders          128.32.0.10 128.32.0.4
```

There are two main reasons for wanting to do so. First, the other systems may not have full network access and may be prevented from sending any IP packets into the rest of the network and, therefore, must rely on a forwarder that does have access to the full net. The second reason is that the forwarder sees a union of all queries as they pass through the forwarder's server and, therefore, the forwarder builds up a very rich cache of data compared to the cache in a typical workstation name server. In effect, the forwarder becomes a meta-cache that all hosts can benefit from, thereby reducing the total number of queries from that site to the rest of the net.

Slave Mode

Slave mode is used if the use of forwarders is the only possible way to resolve queries because of lack of full net access or if you wish to prevent the name server from using other than the listed forwarders. Slave mode is activated by placing the simple command

slave

in the bootfile. If **slave** is used, then you must specify forwarders. When in slave mode, the server forwards each query to each of the forwarders until an answer is found or the list of forwarders is exhausted.

Remote Servers

To set up a host that uses a remote server instead of a local server to answer queries, create the file */etc/resolv.conf*. This file designates the name servers on the network that should be sent queries. It is not advisable to create this file if you have a local server running. If this file exists, it is read almost every time **gethostbyname**(SLIB) or **gethostbyaddr** is called.

Initializing the Cache

The name server needs to know the identities of the authoritative name servers for the root domain of the network. To do this, you have to prime the name server's cache with the address of these higher authorities. This is done in a file called *root.cache*. The location of this file is specified in the boot file */etc/named.boot*.

There are three standard files used to specify the data for a domain. These files are:

named.local
hosts
host.rev.

The *named.local* file specifies the address for the local loopback interface, better known as *localhost*, with the network address 127.0.0.1. The location of this file is specified in the boot file.

The *hosts* file contains all the data about the machines in this zone. The location of this file is specified in the boot file.

The *hosts.rev* file specifies the IN-ADDR.ARPA domain. This is a special domain for allowing address-to-name mapping. Because Internet host addresses do not fall within domain boundaries, this special domain was formed to allow inverse mapping. The IN-ADDR.ARPA domain has four labels preceding it. These labels correspond to the four octets of an Internet address. All four octets must be specified even if an octet is zero. The Internet address 128.32.0.4 is located in the domain 4.0.32.128.IN-ADDR.ARPA. This reversal of the address is awkward to read but allows for the natural grouping of hosts in a network.

Standard Resource Records

The records in the name server data files are called resource records. The following is a general description of a resource record:

```
{name}    {ttl}    addr-class    Record Type    Record Specific data
```

Resource records have a standard format, as shown above. The first field is always the name of the domain record and it must always start in column 1. For some resource records, the name can be left blank. In such cases, the name of the previous resource record is used. The second field is an optional time-to-live field. This specifies how long this data is stored in the database. When this field is left blank, the default time-to-live is specified in the Start of Authority resource record discussed later in this chapter. The third field is the address class. There are currently two classes: IN for internet addresses and ANY for all address classes.

The fourth field states the type of the resource record. The fields after that are dependent on the type of the resource record. Case is preserved in names and data fields when loaded into the name server. All comparisons and lookups in the name server database are case-insensitive.

The following characters have special meanings:

.	A free-standing dot in the name field refers to the current domain.
@	A free-standing @ in the name field denotes the current origin.
..	Two free-standing dots represent the null domain name of the root when used in the name field.
\X	Where X is any character other than a digit (0-9), \X quotes that character so that its special meaning does not apply. For example, "\." can be used to place a dot character in a label.
\DDD	Where each D is a digit, \DDD is the octet corresponding to the decimal number described by DDD. The resulting octet is assumed to be text and is not checked for special meaning.
( )	Parentheses are used to group data that crosses a line. In effect, line terminations are not recognized within parentheses.
;	A semicolon starts a comment; the remainder of the line is ignored.
*	An asterisk signifies a wildcard.

Most resource records have the current origin appended to names if they are not terminated by a ".". This is useful for appending the current domain name to the data, such as machine names, but can cause problems where you do not want this to happen. The following is a good rule of thumb: if the name is not in the domain for which you are creating the data file, end the name with a ".".

Separating Data into Multiple Files

An include line begins with $INCLUDE (starting in column 1) and is followed by a file name. This feature is particularly useful for separating different types of data into multiple files. Here is an example:

```
$INCLUDE /usr/named/data/mailboxes
```

The line would be interpreted as a request to load the file */usr/named/data/mailboxes*. The $INCLUDE command does not cause data to be loaded into a different zone or tree. This is simply a way to allow data for a given zone to be organized in separate files. For example, mailbox data might be kept separately from host data using this mechanism.

Changing an Origin in a Data File

Use the $ORIGIN command to change the origin in a data file. The line starts in column 1 and is followed by a domain origin. This is useful for putting more than one domain in a data file. For example, */etc/named.hosts* might contain lines of the form:

```
$ORIGIN CC.Berkeley.EDU
[assorted domain data...]
$ORIGIN EE.Berkeley.EDU
[assorted domain data...]
```

The Start of Authority Resource Record (SOA)

The Start of Authority record designates the start of a zone. An SOA record includes the following fields:

- Name
- Origin
- Person in charge
- Serial number
- Refresh
- Retry
- Expire
- Minimum

"Name" is the name of the zone. "Origin" is the name of the host on which this data file resides. "Person in charge" is the mailing address for the person responsible for the name server. "Serial number" is the version number of this data file; this number should be incremented whenever a change is made to the data. (Note that the name server cannot handle numbers over 9999 after the decimal point.) "Refresh" indicates how often, in seconds, a secondary name server is to check with the primary name server to see if an update is needed. "Retry" indicates how long, in seconds, a secondary server is to retry after a failure to check for a refresh. "Expire" is the upper time limit, in seconds, that a secondary name server is to use the data before it expires for lack of getting a refresh. Minimum is the default number of seconds to be used for the time-to-live field on resource records. There should only be one SOA record per zone. Here is an example of an SOA record:

```
name  {ttl}  addr-class  SOA     Origin              Person in charge
@            IN          SOA     ucbvax.Berkeley.Edu.  kjd.ucbvax.Berkeley.Edu. (
                         1.1     ; Serial
                         3600    ; Refresh
                         300     ; Retry
                         3600000 ; Expire
                         3600 )  ; Minimum
```


The Name Server Resource Record (NS)

The name server record (NS) lists a name server responsible for a given domain. The first name field lists the domain that is serviced by the listed name server. There should be one NS record for each primary master server for the domain. Here is an example of a name server record:

```
{name}    {ttl}    addr-class    NS    Name servers name
                   IN            NS    ucbarpa.Berkeley.Edu.
```

The address class is IN (Internet addresses), and the record type is name server (NS). The record uses the default ttl (time-to-live) value. Here, the record-specific data is the identity of the name server.

The Address Resource Record (A)

The address record (A) lists the address for a given machine. The name field is the machine name and the address is the network address. There should be one A record for each address

of the machine. Here is an example of an address record for a machine named *ucbarpa* with two network addresses:

```
{name}     {ttl}    addr-class    A    address
ucbarpa             IN            A    128.32.0.4
                    IN            A    10.0.0.78
```

The Host Information Resource Record (HINFO)

The host information resource record (HINFO) is for host-specific data. It lists the hardware and operating system that are running at the listed host. It should be noted that only a single space separates the hardware information and the operating-system information. If you want to include a space in the machine name, you must quote the name. Host information is not specific to any address class, so ANY may be used for the address class. There should be one HINFO record for each host. Here is an example:

```
{name}    {ttl}    addr-class    HINFO    Hardware      OS
                   ANY           HINFO    VAX-11/780    UNIX
```

Note that the current release ignores any records that appear after an HINFO record. Thus, you can use only one HINFO record within the file, and it should be the last record in the file.

The Well-Known Services Resource Record (WKS)

The well-known services record (WKS) describes the well-known services supported by a particular protocol at a specified address. The list of services and port numbers comes from the list of services specified in */etc/services*. There should be only one WKS record per protocol per address. Here is an example of a WKS record:

```
{name}  {ttl} addr-class  WKS   address     protocol  list of services
              IN          WKS   128.32.0.10  UDP       who route timed domain
              IN          WKS   128.32.0.10  TCP       (echo telnet
                                                       discard sunrpc sftp
                                                       uucp-path systat daytime
                                                       netstat qotd nntp
                                                       link chargen ftp
                                                       auth time whois mtp
                                                       pop rje finger smtp
                                                       supdup hostnames
                                                       domain
                                                       name server)
```

The Canonical Name Resource Record (CNAME)

The canonical name resource record (CNAME) specifies an alias for a canonical name. An alias should be the only record associated with the alias name; all other resource records should be associated with the canonical name and not with the alias. Any resource records that include a domain name as their value (for example, NS or MX) should list the canonical name, not the alias. Here is an example of a CNAME record:

```
aliases     {ttl}   addr-class   CNAME   Canonical name
ucbmonet            IN           CNAME   monet
```

The Domain Name Pointer Resource Record (PTR)

A domain name pointer record (PTR) allows special names to point to some other location in the domain. The following example of a PTR record is used in setting up reverse pointers for the special IN-ADDR.ARPA domain. This line is from the example:

```
hosts.rev file.
```

In this record, the name field is the network number of the host in reverse order. You only need to specify enough octets to make the name unique. For example, if all hosts are on network 127.174.14, then only the last octet needs to be specified. If hosts are on networks 128.174.14 and 127.174.23, then the last two octets need to be specified. PTR names should be unique to the zone. Here is an example of a PTR record:

```
name    {ttl}   addr-class   PTR   real name
7.0             IN           PTR   monet.Berkeley.Edu.
```

The Mailbox Resource Record (MB)

The mailbox resource record has a record type of MB. It lists the machine where a user wants to receive mail. The name field is the user's login; the machine field denotes the machine to which mail is to be delivered. Mail box names should be unique to the zone. Here is an example of an MB record:

```
name      {ttl}   addr-class   MB   Machine
miriam            IN           MB   vineyd.DEC.COM.
```

The Mail Rename Resource Record (MR)

The mail rename record (MR) can be used to list aliases for a user. The name field lists the alias for the name listed in the fourth field, which should have a corresponding MB record. Here is an example of a mail rename record:

```
name            {ttl}   addr-class   MR   corresponding MB
Postmistress            IN           MR   miriam
```

The Mailbox Information Resource Record (MINFO)

The mail information record MINFO creates a mail group for a mailing list. This resource record is usually associated with a mail group, but it can be used with a mail box record. The "name" specifies the name of the mailbox. The "requests" field is where mail such as requests to be added to a mail group should be sent. The "maintainer" is a mailbox that should receive error messages. This is particularly appropriate for mailing lists when errors in members' names should be reported to a person other than the sender. Here is an example of this record:

```
name   {ttl}   addr-class   MINFO   requests       maintainer
BIND           IN           MINFO   BIND-REQUEST   kjd.Berkeley.Edu.
```

The Mail Group Member Resource Record (MG)

The mail group record (MG) lists members of a mail group.

```
{mail group name}   {ttl}   addr-class   MG   member name
                            IN           MG   Bloom
```

An example for setting up a mailing list is as follows:

```
Bind        IN   MINFO   Bind-Request              kjd.Berkeley.Edu.
            IN   MG      Ralph.Berkeley.Edu.
            IN   MG      Zhou.Berkeley.Edu.
            IN   MG      Painter.Berkeley.Edu.
            IN   MG      Riggle.Berkeley.Edu.
            IN   MG      Terry.pa.Xerox.Com.
```

The Mail Exchanger Resource Record (MX)

```
name            {ttl}  addr-class  MX  preference value  mailer exchanger
Munnari.OZ.AU.  IN                 MX  0                 Seismo.CSS.GOV.
*.IL.           IN                 MX  0                 RELAY.CS.NET.
```

Mail exchanger records (MX) are used to identify a machine that knows how to deliver mail to a machine that is not directly connected to the network. In the first example above, Seismo.CSS.GOV. is a mail gateway that knows how to deliver mail to Munnari.OZ.AU. but other machines on the network cannot deliver mail directly to Munnari. These two machines may have a private connection or use a different transport medium. The preference value is the order that a mailer should follow when there is more then one way to deliver mail to a single machine. See RFC974 for more detailed information.

Wildcard names containing the character "*" may be used for mail routing with MX records. There are likely to be servers on the network that simply state that any mail to a domain is to be routed through a relay. In the second example above, all mail to hosts in the domain IL is routed through RELAY.CS.NET. This is done by creating a wildcard resource record, which states that *.IL has an MX of RELAY.CS.NET.

Some Sample Files

The following sections contain sample files for the name server. This covers example boot files for the different types of server and example domain database files.

Caching-Only Server

```
;
; Boot file for Caching Only Name Server
;

; type    domain              source file or host
;
domain    Berkeley.Edu
cache     .                   /etc/named.ca
primary   0.0.127.in-addr.arpa   /etc/named.local
```

Primary Master Server

```
;
; Boot file for Primary Master Name Server
;

; type      domain                source file or host
;
directory   /usr/local/lib/named
primary     Berkeley.Edu          ucbhosts
primary     32.128.in-addr.arpa   ucbhosts.rev
primary     0.0.127.in-addr.arpa  named.local
cache       .                     root.cache
```

Secondary Master Server

```
;
; Boot file for Secondary Name Server
;
; type      domain                source file or host
;
directory   /usr/local/lib/named
secondary   Berkeley.Edu          128.32.0.4 128.32.0.10 128.32.136.22 ucbhost.bak
secondary   32.128.in-addr.arpa   128.32.0.4 128.32.0.10 128.32.136.22 ucbhosts.rev.bak
primary     0.0.127.in-addr.arpa  named.local
cache       .                     root.cache
```

The /etc/resolv.conf File

```
domain Berkeley.Edu
name server 128.32.0.4
name server 128.32.0.10
```

root.cache

```
;
; Initial cache data for root domain servers.
;
.                   99999999   IN   NS   SRI-NIC.ARPA.
                    99999999   IN   NS   NS.NASA.GOV.
                    99999999   IN   NS   TERP.UMD.EDU.
                    99999999   IN   NS   A.ISI.EDU.
                    99999999   IN   NS   BRL-AOS.ARPA.
                    99999999   IN   NS   GUNTER-ADAM.ARPA.
                    99999999   IN   NS   C.NYSER.NET.
;  Prep the cache (hotwire the addresses).
SRI-NIC.ARPA.       99999999   IN   A    10.0.0.51
SRI-NIC.ARPA.       99999999   IN   A    26.0.0.73
NS.NASA.GOV.        99999999   IN   A    128.102.16.10
BRL-AOS.ARPA.       99999999   IN   A    128.20.1.2
A.ISI.EDU.          99999999   IN   A    26.3.0.103
BRL-AOS.ARPA.       99999999   IN   A    192.5.25.82
GUNTER-ADAM.ARPA.   99999999   IN   A    26.1.0.13
C.NYSER.NET.        99999999   IN   A    128.213.5.17
TERP.UMD.EDU.       99999999   IN   A    10.1.0.17
```

named.local

```
@     IN    SOA    ucbvax.Berkeley.Edu. kjd.ucbvax.Berkeley.Edu. (
                   1         ; Serial
                   10800     ; Refresh
                   1800      ; Retry
                   3600000 ; Expire
                   86400 )   ; Minimum
      IN    NS     ucbvax.Berkeley.Edu.
1     IN    PTR    localhost.
```

hosts

```
;
;     @(#)ucb-hosts     1.1     (berkeley)     86/02/05
;

@                  IN    SOA     ucbvax.Berkeley.Edu. kjd.monet.Berkeley.Edu. (
                                 1.1        ; Serial
                                 3600       ; Refresh
                                 300        ; Retry
                                 3600000    ; Expire
                                 3600 )     ; Minimum
                   IN    NS      ucbarpa.Berkeley.Edu.
                   IN    NS      ucbvax.Berkeley.Edu.
localhost          IN    A       127.1
ucbarpa            IN    A       128.32.4
                   IN    A       10.0.0.78
                   IN    HINFO   VAX-11/780 UNIX
arpa               IN    CNAME   ucbarpa
ernie              IN    A       128.32.6
                   IN    HINFO   VAX-11/780 UNIX
ucbernie           IN    CNAME   ernie
monet              IN    A       128.32.7
                   IN    A       128.32.130.6
                   IN    HINFO   VAX-11/750 UNIX
ucbmonet           IN    CNAME   monet
```

```
ucbvax          IN   A       10.2.0.78
                IN   A       128.32.10
                IN   HINFO   VAX-11/750 UNIX
                IN   WKS     128.32.0.10 UDP syslog route timed domain
                IN   WKS     128.32.0.10 TCP ( echo telnet
                             discard sunrpc sftp
                             uucp-path systat daytime
                             netstat qotd nntp
                             link chargen ftp
                             auth time whois mtp
                             pop rje finger smtp
                             supdup hostnames
                             domain
                             name server )
vax             IN   CNAME   ucbvax
toybox          IN   A       128.32.131.119
                IN   HINFO   Pro350 RT11
toybox          IN   MX      0  monet.Berkeley.Edu
miriam          IN   MB      vineyd.DEC.COM.
postmistress    IN   MR      Miriam
Bind            IN   MINFO   Bind-Request kjd.Berkeley.Edu.
                IN   MG      Ralph.Berkeley.Edu.
                IN   MG      Zhou.Berkeley.Edu.
                IN   MG      Painter.Berkeley.Edu.
                IN   MG      Riggle.Berkeley.Edu.
                IN   MG      Terry.pa.Xerox.Com.
```

hosts.rev

```
;
;    @(#)ucb-hosts.rev    1.1    (Berkeley)    86/02/05
;

@       IN   SOA   ucbvax.Berkeley.Edu. kjd.monet.Berkeley.Edu. (
                   1.1      ; Serial
                   10800    ; Refresh
                   1800     ; Retry
                   3600000  ; Expire
                   86400 )  ; Minimum
        IN   NS    ucbarpa.Berkeley.Edu.
        IN   NS    ucbvax.Berkeley.Edu.
4.0     IN   PTR   ucbarpa.Berkeley.Edu.
6.0     IN   PTR   ernie.Berkeley.Edu.
7.0     IN   PTR   monet.Berkeley.Edu.
10.0    IN   PTR   ucbvax.Berkeley.Edu.
6.130   IN   PTR   monet.Berkeley.Edu.
```

Additional Sample Files

The following sections contain an additional set of sample files for the name server.

named.boot

```
;
; Name Server boot file for Domain sco.COM
;
; Type        Domain        Source file or Host
;
domain        sco.COM
primary       sco.COM       /etc/named.data/sco-hosts
cache         .             /etc/named.data/root.cache
secondary     sco.COM       /etc/named.data/sco-host.s.rev
primary       sco.COM       /etc/named.data/named.local
```

root.cache

```
;
; Initial cached data for root domain servers.
;
;.                  99999999 IN NS      USC-ISIB.ARPA.
;                   99999999 IN NS      BRL-AOS.ARPA.
;                   99999999 IN NS      SRI-NIC.ARPA.
;
; Insert your own name servers here
;
;                   99999999 IN NS      scovert.sco.COM
;
; Prep the cache (hotwire the addresses)
;
tandy.sco.COM.              99999999 IN A192.9.200.2
;viscous.sco.COM.   99999999 IN A128.0.21.6
;
; Root servers go here
;
tandy.sco.COM.              99999999 IN A192.9.200.2
;SRI-NIC.ARPA.              99999999 IN A10.0.0.51
;USC-ISIB.ARPA.             99999999 IN A10.3.0.52
;BRL-AOS.ARPA.              99999999 IN A128.20.1.2
;BRL-AOS.ARPA.              99999999 IN A192.5.22.82
```

named.local

```
;
; Don't forget to increment the serial number in
; named.soa
;

$INCLUDE /etc/named/sco.soa
192.9.200.2   IN PTR localhost.
```

sco-host.s.rev

```
;
; Don't forget to increment the serial number in
; named.soa
;

$INCLUDE /etc/named/sco.soa

192.9.200.1   IN     PTR    merlin
192.9.200.2   IN     PTR    tandy
192.9.200.3   IN     PTR    tvi
```


sco.soa

```
;
; Don't forget to increment the serial number when you
; change this.  SCCS or RCS might be a good idea here.
;

@             IN SOA  tandy.sco.COM.    root.tandy.sco.COM.  (
                      1.0      ; Serial
                      3600     ; Refresh
                      300      ; Retry
                      3600000  ; Expire
                      3600 )   ; Minimum
              IN NS   tandy.sco.COM.
```

Domain Management

This section contains information for starting, controlling, and debugging **named**(ADMN), the Internet domain name server.

Starting the Name Server

The host name should be set to the full domain style name (that is, monet.Berkeley.EDU.) using **hostname**(TC). The name server is started automatically if the configuration file */etc/named.boot* is present. Do not attempt to run **named** from **inetd**(ADMN). This continuously restarts the name server and defeats the purpose of having a cache.

/etc/named.pid

When **named** is successfully started, it writes its process ID into the file */etc/named.pid.* This is useful to programs that want to send signals to **named.** The name of this file can be changed by defining PIDFILE to the new name when compiling **named.**

/etc/hosts

The **gethostbyname** library call can detect whether **named** is running. If it is determined that **named** is not running, it looks in */etc/hosts* to resolve an address. This option was added to allow **ifconfig**(ADMN) to configure the machine's local interfaces and to enable a system manager to access the network while the system is in single-user mode. It is advisable to put the local machine's interface addresses and a couple of machine names and addresses in */etc/hosts*, so the system manager can copy files from another machine with **rcp** when the system is in single-user mode. The format of */etc/hosts* has not changed. See **hosts**(SFF) for more information. Because the process of reading */etc/hosts* is slow, it is not advisable to use this option when the system is in multiuser mode.

Reload

There are several signals that can be sent to the **named** process to have it do tasks without restarting the process. The SIGHUP signal causes **named** to read *named.boot* and reload the database. All previously cached data is lost. This is useful when you have made a change to a data file and you want **named** 's internal database to reflect the change.

Debugging

When **named** is running incorrectly, look first in */usr/adm/syslog* and check for any messages logged by **syslog**. Next, send it a signal to see what is happening.

SIGINT dumps the current database and cache to */usr/tmp/named_dump.db* This should give you an indication as to whether the database was loaded correctly. The name of the dump file can be changed by defining DUMPFILE to the new name when compiling **named**.

NOTE: The following two signals only work when **named** *is built with DEBUG defined.*

SIGUSR1 - Turns on debugging. Each following USR1 increments the debug level. The output goes to */usr/tmp/named.run.* The name of this debug file can be changed by defining DEBUGFILE to the new name before compiling **named.**

SIGUSR2 - Turns off debugging completely.

For more detailed debugging, define DEBUG when compiling the resolver routines into */usr/lib/libsocket.a.*

Chapter 4
Synchronizing Network Clocks

The clock synchronization service is composed of a collection of time daemons (**timed**(ADMN)) running on the machines in a local-area network. The algorithms implemented by the service are based on a master-slave scheme. The time daemons communicate with each other using the Time Synchronization Protocol (TSP), which is built on the DARPA UDP protocol.

A time daemon has a two-fold function. First, it supports the synchronization of the clocks of the various hosts in a local-area network. Second, it starts (or takes part in) the election that occurs among slave time daemons when, for any reason, the master disappears. The synchronization mechanism and the election procedure employed by the program **timed** are described in the manual page **timed**(ADMN). This chapter is mainly concerned with the administrative and technical issues of running **timed** at a particular site. The next section is a brief overview of how the time daemon works.

How a Time Daemon Works

A master time daemon measures the time differences between the clock of the machine on which it is running and those of all other machines on its network. The master computes the network time as the average of the times provided by nonfaulty clocks. (A clock is considered to be faulty when its value is more than a small specified interval apart from the majority of the clocks of the other machines.) The master time daemon then sends to each slave time daemon the correction that should be performed on the clock of its machine. This process is repeated periodically.

Because the correction is expressed as a time difference rather than an absolute time, transmission delays do not interfere with the accuracy of the synchronization. When a machine comes up and joins the network, it starts a slave time daemon that asks the master for the correct time and resets the machine's clock before any user activity can begin. The time daemons are thus able to maintain a single network time in spite of the drift of clocks away from each other. The present implementation is capable of keeping processor clocks synchronized to within 20 milliseconds, but some hardware is not adjustable at less than 1 second intervals.

To ensure that the service provided is continuous and reliable, it is necessary to implement an election algorithm to elect a new master should the machine running the current master crash, the master terminate (for example, because of a runtime error), or the network be partitioned. Under this algorithm, slaves are able to realize when the master has stopped functioning and to elect a new master from among themselves. It is important to note that the failure of the master results only in a gradual divergence of clock values; thus, the election need not occur immediately.

The machines that are gateways between distinct local-area networks require particular care. A time daemon on such machines may act as a "submaster." This artifact depends on the current inability of transmission protocols to broadcast a message on a network other than the one to which the broadcasting machine is connected. The submaster appears as a slave on one network and as a master on one or more of the other networks to which it is connected.

A submaster classifies each network as one of three types. A slave network is a network on which the submaster acts as a slave. There can only be one slave network. A master network is a network on which the submaster acts as a master. An ignored network is any other network that already has a valid master. The submaster tries periodically to become master on an ignored network, but gives up immediately if a master already exists.

Guidelines

While the synchronization algorithm is quite general, the election algorithm, which requires a broadcast mechanism, puts constraints on the kind of network on which time daemons can run. The time daemon works only on networks with broadcast capability augmented with point-to-point links. Machines that are only connected to point-to-point, non-broadcast networks cannot use the time daemon.

If submasters are excluded, there is normally only one master time daemon in a local-area internetwork. During an election, only one of the slave time daemons becomes the new master. Not all machines are suitable as masters; some do not have sufficiently accurate timing mechanisms or cannot afford the extra overhead. Therefore, a subset of machines must be designated as potential master time daemons. A master time daemon requires CPU resources proportional to the number of slaves (in general, more than a slave time daemon), and so it may be advisable to limit master time daemons to machines with more powerful processors or lighter loads. Also, machines with inaccurate clocks should not be used as masters. This is a purely administrative decision; an organization may well allow all of its machines to run master time daemons.

At the administrative level, a time daemon on a machine with multiple network interfaces may be told to ignore all but one network or to ignore one network. This is done with the **timed -n** *network* and **-i** *network* options, respectively, at startup time. Typically, the time daemon would be instructed to ignore all but the networks belonging to the local administrative control.

There are some limitations to the current implementation of the time daemon. It is expected that these limitations will be removed in future releases. The constant NHOSTS in */usr/src/etc/timed/globals.h* limits the maximum number of machines that can be directly controlled by one master time daemon. The maximum is (NHOSTS - 1). Currently, the maximum is 99. The constant must be changed and the program recompiled if a site wishes to run **timed** on a larger network.

In addition, there is a pathological situation to be avoided at all costs. This situation can occur when time daemons run on multiply-connected local-area networks. In this case, time daemons running on gateway machines are submasters, and they act on some of those networks as master time daemons. Consider machines A and B that are both gateways between networks X and Y. If time daemons were started on both A and B without constraints, it would be possible for submaster time daemon A to be a slave on network X and the master on network Y, while submaster time daemon B would be a slave on network Y and the master on network X. This loop of master time daemons does not function properly or guarantee a unique time on both networks, and it causes the submasters to use large amounts of system resources in the form of network bandwidth and CPU time. In fact, this kind of loop can also be generated with more than two master time daemons, when several local-area networks are interconnected.

Options

The options for the **timed** command are:

-n *network*	Considers the named network.
-i *network*	Ignores the named network.
-t	Places tracing information in */usr/adm/timed.log*.
-M	Allows this time daemon to become a master. A time daemon run without this option is forced into the state of slave during an election.

Daily Operation

The **timedc(ADMN)** command is used to control the operation of the time daemon. It can be used to do the following:

- measure the differences between machines' clocks
- find the location where the master **timed** is running
- cause election timers on several machines to expire at the same time
- enable or disable tracing of messages received by **timed**

See the manual pages on **timed**(ADMN) and **timedc**(ADMN) for more detailed information.

The **rdate(ADMN)** command can be used to set the network date.

Chapter 5
Configuring NFS

SCO ODT-NET Network File System (NFS) is a software product that allows you to mount directories across the network and then treat remote files as if they were local. NFS was developed by Sun Microsystems as a standard for the exchange of data between different machines and operating systems. With the assistance of Lachman Associates, SCO implemented NFS on the UNIX operating system.

As a system administrator, you are responsible for configuring NFS, solving network problems, adding new machines, and adding new users. To assist you in these tasks, the UNIX system provides a convenient set of maintenance commands. Along with the UNIX system utilities, some additional utilities are provided for NFS administration.

Role of the Operating System in NFS

Unlike many recently marketed and distributed operating systems, the UNIX system was originally designed without knowledge that networks existed. This "networking ignorance" presents several impediments to linking the UNIX system with currently available high-performance networks:

- The UNIX system was never designed to yield to a higher authority (like a network-authentication server) for critical information or services. As a result, some operating system semantics are hard to maintain over the net. For example, it may not always be appropriate to trust user ID 0 (*root*).

- The execution semantics in the UNIX system is difficult. For example, these operating systems allow a user to remove an open file, yet the file does not disappear until closed by everyone. In a network environment, a client UNIX machine may not own an open file. Therefore, a server may remove a client's open file.

- When a machine running the UNIX system halts operation, it takes all its applications down with it. When a network node halts for any reason (whether client or server), it should not bring down all of its bound neighbors. The treatment of node failure on a network raises difficulties in any system and is especially difficult in the UNIX environment. A system of *stateless* protocols were implemented to circumvent the problem of a halting server dragging down its bound clients. In this context, *stateless* means that a client is independently responsible for completing work, and that a server need not remember anything from one call to the next. In other words, the server keeps no state. With no state left on the server, there is no state to recover when the server halts and comes back up. From the client's point of view, a halted server appears no different from a very slow server.

In implementing the UNIX system over the network, System V NFS remains compatible whenever possible. However, the following decisions have introduced incompatibilities:

- A preference for making a networked UNIX system a collection of network services rather than having it evolve into a distributed operating system
- A preference for making system halt recovery easier, from both the implementation and the administration points of view

Introducing NFS

In NFS, a server exports directories to be shared and clients mount the directories to access the files in them. The following subsections describe the general interaction of files and daemons in these activities. For details of what happens when a client mounts a remote filesystem, see the section "Remote Mount Failed" later in this chapter.

How Files Interact

To export a directory, the server must list it along with access rights in its */etc/exports* file. In addition, the server must have a name and address for each client to receive services in its */etc/hosts* file.

To mount a directory, the client must have its */etc/default/filesys* file contain all the remote filesystems and locations through which it can access directories. Any **mount**(NADM) command automatically adds an entry to the client's */etc/mnttab*(F) file, and any **umount** command automatically deletes its entry in this file, so this file shows the currently mounted directories.

How Daemons Interact

Two remote programs implement the NFS service — **mountd**(NADM) and **nfsd**(NADM). A client's **mount** request invokes **mountd**, which checks the server's */etc/exports* file for access permission of the client and returns a pointer to a filesystem. After **mount** completes, access to that mount point directory and below goes through the **nfsd** daemon on the server system. Client kernel file access requests (delayed-write and read-ahead) are handled by the **biod** (see **nfsd**(NADM)) daemons on the client.

Setting Up an NFS Client

Setting up an NFS client involves checking the */etc/default/filesys* file and mounting remote filesystems. All examples in this section use the client named *earth*.

Checking the /etc/default/filesys File

A client uses its */etc/default/filesys* file to keep track of specific remote filesystems and directory locations, called *mount points*, through which the client can access directories. You need to make sure that all filesystem information, including access rights, appears in this file. If any information is not present, edit the file to add it. You also need to use **mkdir** to create any mount points that do not already exist in the filesystem.

Here is a sample file:

```
bdev=jupiter:/usr/man mountdir=/usr/man \
fsck=no rcfsck=no rcmount=yes \
mount=no fstyp=NFS nfsopts="soft,rsize=1024,wsize=1024"
```

This entry defines the following:

- The remote resource to mount is jupiter:/usr/man (bdev= entry)
- This resource should be mounted on /usr/man (mountdir= entry)
- It should not be checked by **fsck**(ADM) (fsck= entry)
- It should not be checked by **fsck** when being mounted during normal startup procedures (rcfsck= entry)

- It should be mounted automatically when this system is brought into multiuser mode (rcmount= entry)
- Users should not be able to mount this filesystem on demand (mount=no entry)
- This is an NFS filesystem (fstyp= entry), and that there are some NFS specific option to be given to mount(NADM).

See **filesys**(F) for a description of the syntax and contents of */etc/default/filesys*.

Mounting the Remote Filesystem

The following text replaces the section of the same name.

Any exported filesystem can be remote mounted onto a machine, as long as its server can be reached over the network and the machine is included in the */etc/exports* list for that filesystem. On the machine where the filesystem is to be mounted, the superuser executes the **mount** command. For example, to mount the manual pages from remote machine *jupiter* on the directory */usr/jupiter.man*, enter:

```
mount -r -f NFS,soft jupiter:/usr/man /usr/jupiter.man
```

This example illustrates the use of the **soft** option to specify a soft mount, which means that the client will not retry the mount operation if the server does not respond to the first request. The default is a hard mount, which means that the client continues to try to mount the requested directory even if the server does not respond. A soft mount is recommended for read-only directories, such as online manual pages. The underlying reasoning is that with a soft mount, halting the server does not halt the client.

To make sure the filesystem is mounted where it is expected to be, use the **mount** command without any arguments. This displays the currently mounted filesystems.

Starting and Stopping NFS

NFS must be started by a shell script invoked by **init**(M) when the system is brought up to multi-user mode (run level 2). This is normally configured automatically during installation by linking the script */etc/nfs* to */etc/rc2.d/S89nfs* You need to start NFS on at least one server and at least one of each server's clients to have a functional network.

To stop NFS on a node, log in as *root* and enter the following command:

```
# /etc/nfs stop
```

When you stop NFS on a node, its resources cannot be accessed from any other node. NFS is shut down automatically when the system is taken to run levels 0, 1, or 6.

Debugging NFS

Most problems involving System V NFS network services lie in the one of the following four areas, which are listed in order of probability:

1. The network-access control policies do not allow the operation, or architectural constraints prevent it.
2. The client software or environment is not functioning correctly.
3. The server software or environment is not functioning correctly.
4. The network is not functioning correctly.

To debug NFS, you should be familiar with how NFS works and the names and functions of the various daemons and database files:

biod (see **nfsd**(NADM))	NFS daemon
mount(NADM)	mounts a filesystem
mountd(NADM)	NFS mount requested server
nfsd(NADM)	NFS daemon
rpcinfo(NADM)	reports RPC information
showmount(NADM)	shows all remote mounts
filesys(F)	format of filesystem mount commands table
mnttab(NF)	format of mounted filesystem table
exports(NF)	NFS filesystem being exported

For example, consider a sample mount request as made from an NFS client machine:

```
# mount -f NFS krypton:/usr/src /krypton.src
```

The example asks the server machine *krypton* to return a file handle (**fhandle**) for the directory */usr/src*. This **fhandle** is then passed to the kernel in the **mount**(NS) system call. The kernel looks up the directory */krypton.src* and, if everything is correct, ties the **fhandle** to the directory in a mount record. From now on, all filesystem requests to that directory and below go through the **fhandle** to the server *krypton*.

This describes the way the system should work. We now look at the things that can go wrong: first, some general pointers; then, a list of the possible errors and what might have caused them.

General Hints

When there are network or server problems, programs that access hard-mounted remote files fail in different ways from those that access soft-mounted remote files. Hard-mounted remote filesystems cause programs to retry until the server responds again. Soft-mounted remote filesystems return errors after trying for a while.

Once a hard mount succeeds, programs that access hard-mounted files will hang as long as the server fails to respond. In this case, NFS displays the following message on the console:

```
server not responding
```

On a soft-mounted filesystem, a program receives an error message when it tries to access a file whose server is dead.

If a client is having NFS trouble, first check that the server is up and running. From a client, type the following command to see if the server is up at all:

```
# rpcinfo -p server_name
```

It should display a list of program, version, protocol, and port numbers similar to the following:

```
program vers proto   port
 100000    2   tcp    111  portmapper
 100000    2   udp    111  portmapper
 100017    1   tcp   1024  rexd
 100005    1   udp   1027  mountd
 100003    2   udp   2049  nfs
 100024    1   udp   1039  status
 100024    1   tcp   1025  status
 100021    1   tcp   1026  nlockmgr
 100021    1   udp   1051  nlockmgr
 100020    1   udp   1054  llockmgr
 100020    1   tcp   1028  llockmgr
 100021    2   tcp   1032  nlockmgr
```

You can also use **rpcinfo** to check if the **mountd** server is running:

```
# rpcinfo -u server_name mountd
```

This should return:

```
program 100005 version 1 ready and waiting
```

If these steps fail, try to log in on the server's console to see if it is running.

If the server is alive but a client machine cannot reach it, check the Ethernet connections between the machines.

If the server and the network are alive, use **ps** to check the client daemons. A **nfsclnt** and several **biod** daemons should be running. For example, the command:

```
# ps -ef
```

should print lines for **nfsclnt** and **biod**.

The following sections deal with the most common types of failure. The section "Remote Mount Failed" covers the steps to take if a remote mount fails. The sections "Programs Are Hung" and "Everything Works Slowly" discuss servers that do not respond when filesystems are mounted.

Remote Mount Failed

This section deals with problems related to mounting remote filesystems. If **mount** fails for any reason, check the sections below for specific details about what to do. They are arranged according to where they occur in the mounting sequence and labeled with the error message likely to be displayed.

The **mount** command can get its parameters either from the command line or from the file */etc/default/filesys*. (See **mount**(NADM).) The example below assumes command-line arguments, but the same debugging techniques would apply if */etc/default/filesys* were the source of the options.

The interaction of the various parts of the mount request should be considered. In the example mount request given above:

mount -f NFS krypton:/usr/src /krypton.src

mount goes through the following steps to mount a remote filesystem.

1. The **mount** command opens */etc/mnttab* and checks that this mount was not already done.

2. The **mount** command parses the first argument into host *krypton* and remote directory */usr/src*.

3. The **mount** command then resolves the host *krypton* into an Internet protocol address.

4. The **mount** command calls *krypton*'s portmapper to get the port number of **mountd**.

5. The **mount** command calls *krypton*'s **mountd** and passes it to the */usr/src* pathname.

6. *krypton*'s **mountd** reads its */etc/exports* and looks for the exported filesystem that contains */usr/src*.

7. *krypton*'s **mountd** does a **nfs_getfh**(NS) system call on */usr/src* to get the **fhandle**.

8. *krypton*'s **mountd** returns the **fhandle** to the client.

9. On the client machine, **mount** does a **mount**(NS) system call with the **fhandle** and */krypton.src*.

10. The **mount** command checks whether or not the caller is a superuser and */krypton.src* is a directory.

11. The **mount** command does a **statfs**(S) call to *krypton*'s NFS server (**nfsd**).

12. The **mount** command opens */etc/mnttab* and adds an entry.

Any one of these steps can fail, and some of them can fail in more than one way. The following sections give detailed descriptions of the failures associated with specific error messages.

`mount: can't open /etc/mnttab`

The table of mounted filesystems is kept in the file */etc/mnttab*. This file must exist before **mount** can succeed. The file */etc/mnttab* is created when the system is booted, and it is maintained automatically after that by the **mount** and **umount** commands.

`mount: /dev/nfsd is already mounted,` *sys_name* `is busy, or allowable number of mount points exceeded`

This message reveals an attempt to mount a filesystem that is already mounted. All NFS mount requests that fail with this message display the name */dev/nfsd* (a byproduct of the implementation) regardless of the actual mount request.

`mount:` *name* `or` *name*`, no such file or directory`

The **-f NFS** or **krypton:** part of the sample command was probably omitted. The **mount** command assumes that a local mount is being done unless the **-f** flag is used on the command line, or the requested directory as listed in */etc/default/filesys* specifies filesystem type **NFS**.

More simply, this message also appears when the specified local mount point is not an existing directory.

`mount: can't open /etc/default/filesys`

The **mount** command tried to look up the information needed to complete a mount request in */etc/default/filesys*, but there was no such file. As the system administrator, you need to create this file as part of initial system setup.

sys_name `not in hosts database`

The system name specified on the **mount** command suffixed by the ":" is not listed in the file */etc/hosts*. Check the spelling of the host name and the placement of the colon in the **mount** command.

`mount: directory argument` *name* `must be a full pathname`

The second argument to **mount** is the path of the directory to be covered. This must be an absolute path starting at slash (/).

`mount: ... server not responding(1): RPC_PMAP_FAILURE - RPC_TIMED_OUT`

Either the server to which the mount is being attempted is down, or its portmapper is dead or hung. Attempt to log in to that machine; if that attempt succeeds, then the problem may be in the portmapper. Run the following command from your system as the superuser to test the portmapper on the server system:

rpcinfo -p *hostname*

The result should be a list of registered programs. If this is not the case, kill the remote portmapper and restart it. Restarting the portmapper is a complicated process because all registered services are lost, and their associated daemons must be restarted, too. To find the process IDs of **portmap** and other service daemons, enter this command as the superuser:

ps -ef

Kill the daemons with:

kill -9 *portmap_pid daemon_id1 daemon_id2*

Start new ones with commands such as:

```
# /etc/portmap
# /etc/mountd
# /etc/nfsd
```

Another alternative to all this is simply to reboot the server when it is convenient. Because of the stateless nature of the NFS server implementation, there should be no adverse effect on the clients of the system, other than the time that they suspend awaiting the return of the server.

If the server is up but it is not possible to **rlogin** to it, check the client's Ethernet connection by trying to **rlogin** to another machine, or to ping another machine. Also, check the server's Ethernet connection.

`mount: ... server not responding: RPC_PROG_NOT_REGISTERED`

This means that **mount** got through to the portmapper, but the NFS mount daemon **mountd** was not registered. The server should be checked to ensure that */etc/mountd* exists and is running.

`mount: /dev/nfsd or` *name*`, no such file or directory`

Either the remote directory does not exist on the server or the local directory does not exist. Again, note that */dev/nfsd* is always printed to represent the remote directory.

`mount: access denied for` *sys_name*`:`*name*

The client machine on which the mount attempt is being made is not in the server's export list for the filesystem to be mounted. A list of the server's exported filesystems can be obtained by running:

```
# showmount -e hostname
```

If the desired filesystem is not in the list, or the machine name is not in the user list for the filesystem, then check the */etc/exports* file on the server for the correct filesystem entry. A filesystem name that appears in the */etc/exports* file but not in the output from **showmount** indicates a failure in **mountd**. Perhaps it could not read that line in the file, or it could not find the filesystem, or else the filesystem name was not a locally mounted filesystem. See *exports*(NF) for more information.

This message can also be an indication that authentication failed on the server. It may be displayed because the machine that is attempting the mount is not in the server's export list, because the server is not aware of the machine, or because the server does not believe the identity of the machine. Check the server's */etc/exports* file.

`mount: `*`name`*`: no such file or directory`

The remote path on the server is not a directory.

`mount: not superuser`

The **mount** command can be used only by the superuser, because it affects the filesystem for the entire machine.

Programs Are Hung

If programs hang doing file-related work, the NFS server may be dead. The following message may be displayed on the machine's console:

`NFS server `*`sys_name`*` not responding, still trying`

The message includes the name of the NFS server that is down.

This is probably a problem either with one of the NFS servers or with the Ethernet.

If a client machine completely stops responding, check the servers from which the filesystems were mounted. If one of them is down, client machines may wait indefinitely. When the server comes back up, programs continue automatically and are unaffected.

If a soft-mounted server dies, other work should not be affected. Programs that time-out trying to access soft-mounted remote files will fail, but it should still be possible to get work done on other filesystems.

If other clients of the server seem to be functioning correctly, the Ethernet connection and the connection of the server should be checked.

Everything Works Slowly

If access to remote files seems unusually slow, the server should be checked by entering (on the server):

```
# ps -ef
```

If the server is functioning and other users are getting good response, block I/O daemons on the client should be checked with the command **ps -ef** (on the client) and looking for **biod**. If the daemons are not running or are hung, they should be killed. First, find the process IDs by entering:

```
# ps -ef | grep biod
```

Then, kill them with:

```
# kill -9 pid1 pid2 pid3 pid4
```

Restart the daemons with:

```
# /etc/biod 4
```

To determine whether the daemons are hung, use **ps** as above, then copy a large file. Another **ps** will show whether the **biod**s are accumulating CPU time; if not, they are probably hung.

If **biod** appears to be functioning correctly, check the Ethernet connection. Use the **nfsstat -c** and **nfsstat -s** commands to discover whether a client is doing a lot of retransmitting. A retransmission rate of 5% is considered high. Excessive retransmission usually indicates a bad Ethernet board, a bad Ethernet tap, a mismatch between board and tap, or a mismatch between the client machine's Ethernet board and the server's board.

Serial Number Validation Errors

When you see a message of the following nature:

```
Copy Protection Violation!
Duplicate Serial Number(s) detected on 123.4.567.007
Product(s) att00001 shutting down.
```

a validation error has occurred. This means that there are two copies of the same program running at the same time. You must remove one of the copies.

Adding a New User

Adding a new user involves adding an entry to the proper password files and, for local logins, creating a home directory on the new user's machine. For general information on how to do this, see *Administering ODT-OS* in the *Administrator's Guide*.

Within a network environment, to add a new user, you should add a password file entry to every machine on the local network. Each user and group must have a unique identification number across the entire network, rather than just on the home machine. A server machine must have an entry for each user on each client machine who will be using NFS services.

The user ID should be a number unique to the user. A system knows the user by the ID number associated with the login name; therefore, a login name must have the same user ID number in all password files of machines that are networked in a local domain. Failure to keep IDs unique will prevent users from moving files between directories on different machines, because the system will respond as if the directories were owned by two different users. In addition, file ownership may become confused when an NFS server exports a directory to an NFS client whose password file contains users with UIDs matching those of different users on the NFS server.

Incompatibilities with Remote Filesystems

A few things work in different ways, or not at all, on remote NFS filesystems. The next section discusses the incompatibilities and offers suggestions for working around them.

No SU over the Network

Under NFS, a server exports filesystems it owns, so that clients may remotely mount them. When a user on a client machine becomes the superuser, it is denied superuser permission on remote-mounted filesystems. Consider the following actions performed by the user *jsbach* on the server machine:

```
$ cd
$ touch test1 test2
$ chmod 777 test1
$ chmod 700 test2
$ ls -l test*
-rwxrwxrwx  1 jsbach       0 Mar 21 16:12 test1
-rwx------  1 jsbach       0 Mar 21 16:12 test2
```

The superuser on the client machine tries to access the remote filesystem on the server machine, intending to use the superuser privileges:

```
$ su
Password:
# cd /usr2/jsbach
# touch test1
# touch test2
touch: test2: Permission denied
# ls -l test*
-rwxrwxrwx  1 jsbach       0 Mar 21 16:16 test1
-rwx------  1 jsbach       0 Mar 21 16:12 test2
```

When checking permissions for accessing a remote-mounted filesystem, the server considers the superuser from a client machine to be in the "other" category.

The problem usually shows up during the execution of a setuid *root* program. Programs that run as *root* cannot access files or directories unless the permission for "other" allows it. When operating from a client machine, the uid for root is mapped to -2 (or 65534).

Another aspect of this problem is that ownership of remote-mounted files sometimes cannot be changed, specifically if they are on a server that does not permit users to execute **chown**. Because *root* is treated as the "other" user for remote accesses, only *root* on the server can change the ownership of remote files. For example, consider a user trying to **chown** a new program *a.out* that must be setuid to *root*. It does not work, as shown below:

```
$ chmod 4777 a.out
$ su
Password:
# chown root a.out
a.out: Not owner
```

To change the ownership, you must either log in to the server as *root* and make the change, or move the file to a filesystem owned by the user's machine (for example, */usr/tmp*, which is usually owned by the local machine) and make the change there.

File Operations Not Supported

File locking of directories is not supported on remote filesystems.

In addition, Append mode and atomic writes are not guaranteed to work on remote files accessed by more than one client simultaneously.

Cannot Access Remote Devices

With NFS, it is not possible to access a remote-mounted device or any other character, block-special file, or named pipes.

Cannot Access Different Filesystem Types

Under NFS, it is not possible to access a filesystem of a type different from that of your native system. For example, you are running NFS under UNIX. It is not possible to access a DOS filesystem from your UNIX client machine.

Cannot Access Indirect Filesystems

Under NFS, it is not possible to access a filesystem indirectly. For example, you may not use a server machine to access a filesystem on a third server machine. All NFS access must be direct from server to client.

Handling Clock Skew in User Programs

Because the NFS architecture differs in some minor ways from earlier versions of the XENIX and UNIX systems, users should be aware of those places where their own programs could run up against these incompatibilities.

The clocks on the NFS server and client may be out of sync because each machine keeps its own time. This might cause problems. Consider the following example.

Many programs assume that an existing file could not be created in the future. For example, the command **ls -l** has two basic forms of output, depending upon how old the file is:

```
$ date
Sat Apr 12 15:27:48 GMT 1986
$ touch file2
$ ls -l file*
-rw-r--r--  1 jsbach          0 Dec 27  1984 file
-rw-r--r--  1 jsbach          0 Apr 12 15:27 file2
```

The first type of output from **ls** prints the year, month, and day of last file modification if the file is more than six months old. The second form prints the month, day, hour, and minute of last file modification if the file is less than six months old.

The **ls** command calculates the age of a file simply by subtracting the modification time of the file from the current time. If the result is greater than six months, the file is "old".

Assume that the time on the server is Apr 12 15:30:31, which is three minutes ahead of the local machine's time:

```
$ date
Apr 12 15:27:31 GMT 1986
$ touch file3
$ ls -l file*
-rw-r--r--  1 jsbach          0 Dec 27  1983 file
-rw-r--r--  1 jsbach          0 Apr 12 15:26 file2
-rw-r--r--  1 jsbach          0 Apr 12  1986 file3
```

The difference between the current time and the library's modify time makes the **ls** command think that the new file was created long ago.

In general, users should remember that applications that depend upon local time and/or the filesystem timestamps will have to deal with clock skew problems if remote files are used.

Chapter 6

Managing the LAN Manager Client Network

Installation of the ODT-NET LAN Manager Client network consists of establishing unique user and group IDs throughout the network and configuring the computer as a consumer. Chapter 1 discusses the importance of establishing consistency of network ID files. The rest of this section describes how to use the **mkself**(XC) command to configure a computer and how to add a new computer to the network. When you install the LAN Manager Client software, the **custom** program performs these operations automatically. The information is provided here as a tool for system maintenance; it is not necessary to duplicate the actions performed by the **custom** program.

The remaining sections of the chapter discuss:

- the special network files used by LAN Manager Client
- starting and stopping the LAN Manager Client network
- NetBIOS
- network parameters
- configuring for performance

This chapter focuses on administering a LAN Manager Client network to connect to a DOS or OS/2 machine on a LAN Manager network. For information about providing access to a XENIX-NET server, see the documentation provided with the XENIX-NET server.

Defining Network Computers: The mkself Utility

The **mkself** command builds the necessary network files on a network computer. The **custom** program performs this operation for all network computers.

From the root account, enter the command:

```
# mkself name
```

where *name* is the network name of the computer. (See Chapter 1 of this guide for computer naming restrictions.) The **mkself** utility edits the */etc/systemid* file to include the machine name.

Adding Network Computers

Once the network is installed, it may be necessary to alter the network's setup to meet changing computing needs and to remedy any problems that arise due to changes in your system.

When adding a new computer to the network, the procedure is essentially the same as for initial installation. Most tasks are performed by the **custom** utility during the installation. If you choose to have the network start automatically, **custom** places the **net start rdr** command in the file */etc/rc.d/6/xnet.6*. This command automatically starts your computer as a network consumer each time you enter multiuser mode. Remember that you must be logged in as the super user to add computers.

1. Choose a name and password for the new machine. The name must not duplicate any name already on the network or any top-level file or directory name. See "Starting and Stopping the Network" later in this chapter for the startup commands.

2. **custom** configures each computer by using the **mkself** command to add the machine name to the file */etc/systemid.*

All consumers must be added as users to the server machine they access.

Special Network Files

The network requires special files to operate. These files allow network commands and requests to locate users, groups, computers, and resources. This section describes each of the special networking files. This section does not explain any files already present in the UNIX operating system that are used by the network. Also, the actual LAN Manager Client program files are not listed as part of this section. The network parameters mentioned here are described in detail under "Network Parameter Descriptions" later in this chapter.

Consumer Configuration File (/usr/lib/xnet/constable)

The *constable* file is used by UNIX system consumers to specify five parameters that affect the performance of data transfer between the consumer and a server. The parameters can be individually specified for different servers. This file does not need to be changed for the normal operation of LAN Manager, although modifications to it can improve network performance.

Network Parameter File (/usr/lib/xnet/xnetrc)

Network parameters can be altered via the file */usr/lib/xnet/xnetrc*. After modifications have been made to this file, the network must be restarted for the changes to take effect.

Starting and Stopping the Network

This section explains how to start the LAN Manager Client network both automatically and by entering commands. It also explains how to stop the network. Remember that you must be logged in as the super user to give these commands.

Starting the LAN Manager Client Network

To start your computer as a consumer, use one of the following commands:

net start rdr	Starts your computer as a consumer.
xfc on	Starts your computer as a consumer.

If, during the installation procedure, you chose to activate LAN Manager Client automatically upon going into multiuser mode, the installation script copies the file *xnet.6* from */usr/lib/xnet/rc.d* to the */etc/rc.d/6* directory. This automatically starts your computer as a

network consumer each time you start your computer in multiuser mode. Once this file is copied into */etc/rc.d/6/xnet.6*, manual entering of the **xfc** or **net start** command is not needed.

If, during the installation procedure, you chose not to have LAN Manager Client start automatically, the file *xnet.6* is not copied to the */etc/rc.d/6* directory. This file is stored in */usr/lib/xnet/rc.d*. If you decide at this time to have LAN Manager Client start automatically, copy this file to */etc/rc.d/6*.

Stopping the LAN Manager Network

If a serious problem occurs with the network software or hardware, it may be necessary to halt the network processes. Normally, users simply log off until the problem is fixed. However, if users can continue their work without using the network, the following commands stop network functions but leave a computer available for local work:

net stop rdr	Stops your computer as a consumer. Existing connections to servers are unaffected. No new network connections are allowed.
xfc off	Shuts down the consumer. Existing connections to servers are unaffected. No new network connections are allowed.
xfc abort	Immediately stops the consumer. All connections to servers are closed. No new network connections are allowed.

NetBIOS

LAN Manager Client operates as a NetBIOS consumer. The maximum and default number of NetBIOS sessions available can vary with the networking hardware being used.

LAN Manager Client filesharing creates and maintains a list of sessions during the course of operation. These sessions can be in the active state, which means that one server process exists on the server side of the session for each client process on the consumer side of the session. If no server-consumer relationship currently exists, this means that either there are no open remote files or that there are no processes with remote current directories. If this is the case, the session is said to be dormant.

LAN Manager Client uses its own limited virtual circuit table. It does not try to use more than the number of virtual circuits specified in the NVCSUSED parameter. If LAN Manager Client needs to communicate with a machine with which it does not currently have a session, it creates a new session. If this causes LAN Manager Client to exceed the size of its virtual circuit table, dormant circuits are recycled. If no circuits are dormant, LAN Manager Client returns an error message.

If you will be using the Open Desktop Development System to build int5c applications, there are some special considerations you should be aware of. These applications use NetBIOS sessions. The NetBIOS sessions used by these applications are in addition to those used by the LAN Manager Client filesharing. Thus, if you wish to use both the LAN Manager Client filesharing and the applications that share the NetBIOS with LAN Manager Client, you must limit the number of NetBIOS sessions that the filesharing attempts to use. This is done by specifying the value of the NVCSUSED parameter in the */usr/lib/xnet/xnetrc* file, in conjunction with the MAXVCS parameter in the */etc/conf/cf.d/mtune* file.

The value of the NVCSUSED parameter must be set to equal the total number of NetBIOS sessions available, less the number of simultaneous sessions to be used by other applications and utilities. Use the **xnstatus** command to determine the number of NetBIOS sessions available to be configured. See the **xnstatus**(XC) manual page for more information about the **xnstatus** command. The number of NetBIOS sessions available can be adjusted to be between one and the maximum number available using **xnreset**.

Network Parameter Descriptions

The network filesharing parameters located in the file */etc/conf/cf.d/mtune* determine the amount of memory allocated to data structures in the kernel. These parameters have an effect only at kernel link time. In contrast, the parameters located in */usr/lib/xnet/xnetrc* determine how LAN Manager Client distributes the network resources among the network machines. The *xnetrc* parameters affect the runtime operation of LAN Manager Client and do not affect the size of the kernel. For example, the NPTE parameter in *mtune* determines the size of the (system-wide) network process table that LAN Manager Client uses. The NPTE parameter keeps track of the processes using network resources. The MSPVC limits the number of processes that can use a particular virtual circuit, even though more spaces may be available in the network process table.

Changing Network Parameters

To change a network parameter located in */etc/conf/cf.d/mtune*, add the parameter to the file */etc/conf/cf.d/stune* and assign it the desired value. When you have finished modifying *stune*, type:

link_unix

to relink the kernel. Network parameter values specified in */etc/conf/cf.d/stune* override those specified in */etc/conf/cf.d/mtune*.

NOTE: It is not necessary to change any of the parameters under normal conditions. Changing any parameter may adversely affect LAN Manager Client's performance.

The following table shows the parameters named in */etc/conf/cf.d/mtune* that you can include and modify in */etc/conf/cf.d/stune*. The default values listed are for machines using the Intel® 80386™ processor.

Descriptions of the parameters follow the tables.

Parameter	Default Value
NNCB_DEV	32
NNCBS	32
NNCB_NAMES	16
NFSTREAM	32
NFSBUFS	60
NBINDX	80
MAXVCS	20
NPTE	64
NBIDS	32
NTIDS	32
NEXS	150
NWB	32
NASEVENT	40
NCONSTABLE	32
NALIAS	16
NRECYCLE	3
NCALLRETRY	10
XITONCLOSE	0
CNCBS	0
CORMAPNCB	0

Streams Parameter	Default Value
NQUEUE	256
NSTREVENT	64
N4K	0
N2K	10
N1K	20
N512	10
N256	10
N128	10
N64	30
N16	60
N4	128

NOTE: These defaults may be different on your system.

CNCBS

Number of co-resident NCB structures allocated.

Default=0

This parameter is set to 0 when only one NetBIOS is installed.

CORMAPNCB

This is a structure used for mapping NCBS to sessions.

Default=0

This parameter is set to 0 when only one NetBIOS is installed.

MAXVCS

Maximum number of virtual circuit table entries.

Default=20

This structure is used by the network filesharing software to keep track of the state of the virtual circuits. The value of this parameter should not exceed the number of virtual circuits supported by the lower layers of the network.

NALIAS

Number of alias table entries.

Default=16

The value for NALIAS assigned in the file */etc/conf/cf.d/mtune* specifies the number of 12-word structures (nc alias) reserved in the kernel data segment. Each time a UNIX system consumer uses the **net use** command, an entry in this table is used. This table keeps track of the mapping of the alias name used to the appropriate virtual circuit. If all the table space is used, the **net use** command returns an error message to the user.

NASEVENT

Number of async event cells allocated in the kernel.

Default=40

One event cell is used for each outstanding read-ahead or write-behind request. If this parameter is set too low, this message is displayed:

```
LAN Manager Client: low on async cells (NASEVENT)
```

To fix this problem, either raise NASEVENT or decrease the read-ahead and write-behind window sizes. For more information on the window sizes, see "Configuring for Performance."

NBIDS

Number of bind table entries.

Default=30

The value for NBIDS assigned in the file */etc/conf/cf.d/mtune* specifies the number of 11-word structures (bidtab) reserved in the kernel data segment. The error message "Server: out of bind entries (NBIDS)" indicates that NBIDS needs to be increased on your network.

NBINDX

Number of BINDX structures.

Default=80

Used for tracking read-ahead requests. This message indicates that NBINDX should be raised or the read-ahead windows should be reduced:

```
LAN Manager Client: low on bindxs (NBINDX)
```

For more information on read-ahead windows, see "Configuring for Performance" later in this chapter.

NCALLRETRY

Number of attempts to establish a session.

Default=10

This is the number of times LAN Manager Client attempts to establish a session if the initial attempt fails. This retry only occurs when certain conditions on the server exist, and it is likely that a retry succeeds.

NCONSTABLE

Number of consumer table structure entries.

Default=32

This parameter corresponds to the number of distinct servers specified in the file */usr/lib/xnet/constable*. If you add more servers, increase this value accordingly.

NEXS

Number of exclude table entries.

Default=150

The value for NEXS assigned in the file */etc/conf/cf.d/mtune* specifies the number of four-word structures (extab) reserved in the kernel data segment. The error message "Server: out of exclude fid entries (NEXS)" indicates NEXS may need to be increased on your network.

NFSBUFS

Maximum number of network buffers.

Default=60

This parameter determines the total number of network buffers available. Reducing this number increases the amount of memory available for user programs.

NFSTREAM

Number of streams to the adapter.

Default=32

One stream is used for each filesharing virtual circuit.

NNCBS

Maximum number of NCBs.

Default=32

NNCB_DEV

Number of sessions the network driver can handle.

Default=32

NNCB_NAMES

Maximum number of names that the network device allows.

Default=16

NPTE

Number of process environment table entries.

Default=64

The value for NPTE assigned in the file */etc/conf/cf.d/mtune* specifies the number of 11-word structures (nptab) reserved in the kernel data segment. These structures are used by the network software to keep track of processes using network resources. An entry in this table is used when a local process first references a remote file. The entry is released when the process exits.

The error message "out of network process table space (npte)" indicates that the value of NPTE is too low for your network activity.

NQUEUE

Number of stream queues.

Default=256

The number of stream queues must be at least four times the value of NFSTREAM, plus all queues used by other software in the kernel.

If you change the value of the NQUEUE parameter, errors can occur.

NRECYCLE

Maximum number of dormant circuits LAN Manager Client recycles.

Default=4

This is the maximum number of dormant circuits that LAN Manager Client recycles when its virtual circuit table is full, and it needs to consume from a new server.

NSTREVENT

Number of stream event cells.

Default=64

NTIDS

Number of tree connect table entries.

Default=32

The value for NTIDS assigned in the file */etc/conf/cf.d/mtune* specifies the number of five-word structures (tidtab) reserved in the kernel data segment. Tree connects bind a consumer alias with a server. The error message "Server: out of tree connect entries (NTIDS)" indicates that the NTIDS value may need to be increased.

NWB

Number of write-behind control structures.

Default=32

One NWB is allocated for each file being written to. This message indicates that more NWBs should be allocated:

```
LAN Manager Client: out of write behind table entries (NWB)
```

XITONCLOSE

Allows virtual circuits to become dormant when all files close.

Default=0

This is a Boolean variable. If XITONCLOSE is set to non-zero, filesharing virtual circuits become dormant when all files are closed. If it is zero, circuits remain active until all processes that use remote resources on the circuit have exited. The advantage to leaving the circuit active is that subsequent opens have less overhead associated with them on the server machine. Applications that frequently open and close many files benefit from this. The advantage to allowing circuits to become dormant is that the server then has the option of recycling the circuit to serve another machine if it becomes necessary.

Number of Stream Data Blocks

N4

Number of stream data blocks of size 4.

Default=60

N16

Number of stream data blocks of size 16.

Default=60

N64

Number of stream data blocks of size 64.

Default=30

N128

Number of stream data blocks of size 128.

Default=10

N256

Number of stream data blocks of size 256.

Default=10

N512

Number of stream data blocks of size 512.

Default=20

N1K

Number of stream data blocks of size 1024.

Default=20

N2K

Number of stream data blocks of size 2048.

Default=12

N4K

Number of stream data blocks of size 4096.

Default=0

Changing Network Filesharing Parameters Located in xnetrc

The network filesharing parameters located in the */usr/lib/xnet/xnetrc* file determine various functional limits in the LAN Manager Client software. The network server reads the *xnetrc* file when filesharing is started with the **net start** command. The filesharing program checks for entries that override default filesharing parameters.

The network system administrator can change some network parameters by modifying the */usr/lib/xnet/xnetrc* file. It is unnecessary to change any of the parameters under normal conditions. Changing any parameters can adversely affect LAN Manager Client performance.

The following table shows the parameters and default values as named in the */usr/lib/xnet/xnetrc* file. Parameter descriptions follow the table.

Parameter	Default Value
NVCSUSED	20
MBPVC	15
MCONVCS	20
MFPVC	500
MSERVCS	20
MTPVC	20
NBIOSIZ	0 (auto-configured)
NETDEV	0
NBUFALLOC	0 (auto-configured)
NORDONLY	0

NVCSUSED

Maximum number of virtual circuits that are used by the network filesharing code.

Default=20

This structure is used by the network filesharing software to keep track of the state of the consumer virtual circuits. The value of this parameter should not exceed the number of virtual circuits supported by the lower layers of the network. It should also not exceed the number specified in MAXVCS in the *mtune* file. Applications written using the int5c library can use virtual circuits other than the ones specified in NVCSUSED.

MBPVC

Maximum number of binds per server virtual circuit.

Default=15

Non-core protocol virtual circuits allow consumers to "bind" to a directory or file. XENIX computers use this process when a user changes to a directory on a remote computer. This parameter limits the number of bind table entries that a single virtual circuit can use.

MCONVCS

Maximum number of virtual circuits for use by the consumer.

Default=20

The value of MCONVCS cannot exceed the value of NVCSUSED.

MFPVC

Maximum number of fids per server virtual circuit.

Default=500

This parameter limits the number of open files allowed at one time by the consumer using a virtual circuit.

MSERVCS

Maximum number of virtual circuits for use by the server.

Default=20

The value of MSERVCS cannot exceed the value of NVCSUSED.

MTPVC

Maximum number of tree connects per virtual circuits.

Default=20

This parameter limits the number of tree connect table entries (tids) that can be used by a single virtual circuit. Tree connects are used to "login" to a server when the server is validating users.

The maximum value equals the value of NTIDS in the *mtune* file.

NBIOSIZ

Size of network buffers.

Default=0

This parameter defines the size of the actual buffer space used to send and receive messages. This parameter is auto-configured if a value of 0 is specified.

For detailed information on configuring the buffer size, see "Configuring for Performance."

NETDEV

Major device number of the network device used for filesharing.

Default=0

NETDEV specifies which networking device filesharing requests are routed to. Normally, the device driver is provided by your network supplier.

NBUFALLOC

Number of network buffers.

Default=0

This parameter specifies the total number of network buffers allocated when filesharing is started. This parameter is auto-configured if a value of 0 is specified.

For detailed information on configuring buffers, see "Configuring for Performance."

NORDONLY

No read-only core protocol file simulation.

Default=0

The core protocol supports file opens that are exclusive to a consumer, but can be shared among processes within that consumer. However, it is often desirable to share such files among other network computers, as long as only read accesses are permitted. If NORDONLY is set for zero, such files will be available across the network for reading. If it is non-zero, no multiple-computer accesses will be allowed on open files.

Configuring for Performance

This section explains how to improve LAN Manager Client's performance by altering the configuration of your system. None of what is described here is required for the normal operation of LAN Manager Client.

Adjusting Consumer Read and Write Windows

By editing the file */usr/lib/xnet/constable* on your consumer machines, you can individually adjust the read and write windows (described below) for each server that the consumer is in contact with. You can greatly enhance network performance by carefully tuning these parameters.

The *constable* file has this format:

```
#              Special                 General                 General
#            Read Support            Read-Ahead              Write-Behind
#
#---------+---------------+-------------------------+----------------+
# server        reado          reada    read    read          write
# name          on/off         timeout  window  ahead         window
#
*               1              500      3       3             4
```

Note that every line in the table except the last one is commented out. This line has an asterisk (*) in the "server name" column, and it lists the default values for each parameter. To specify different values for a specific server, add a line to the bottom of the file that has the server's name at the beginning of the line, and the values you want to use for that server in the other columns. Any machine that does not have a line of its own uses the defaults specified on the line beginning with the asterisk.

The sections below describe the parameters and suggest how to change them to improve performance.

reado on/off

This parameter is a toggle. A value of 1 turns on an improved protocol set, and a value of 0 turns it off. It should always be set to 1.

reada timeout

This sets the length of time that buffered data remains valid, in units of 1/50th of a second. For example, if this parameter is set to 500, data that was read from a server and placed in a buffer remains valid for 10 seconds. If the data is not read from the buffer within this period, the buffer is flushed and the data is re-read from the server.

Setting this parameter to a high number improves network performance, but it can cause data integrity problems if the data being read changes frequently. As a general guideline, use a high value for machines that contain relatively stable data (such as archive servers) and a lower value for machines with more volatile data.

read window

With this parameter, you can specify the number of read requests that may remain outstanding before a reply is received. A value of zero or one causes the read requests to be sent one at a time; after a request is sent, no other requests are sent until a reply has been received. A value of 2 allows two requests to be sent before a reply is received. Figure 6-1 illustrates this difference:

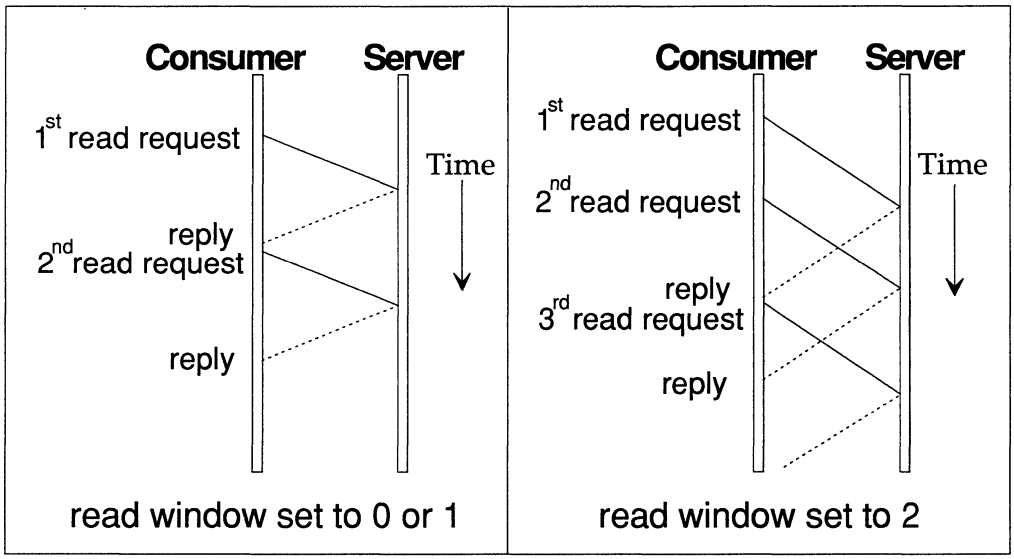


Figure 6-1. The read window size affects performance.

Setting this parameter to a high value dramatically improves network performance, but it is a drain on system resources. You may wish to test several different values to find what works best on your system.

read ahead

The read-ahead value is the number of extra read requests that LAN Manager Client sends after the initial request was satisfied. By setting this parameter to a value greater than 0, you allow LAN Manager Client to anticipate a user process by requesting data that the process has not yet asked for.

A high read-ahead value improves network performance, but it increases the drain on system resources. As with the read window, the best way to find the ideal value is through experimentation.

write window

The write window is very similar to the read window. This parameter specifies the number of write requests that may be outstanding before the server acknowledges that the data was received intact. If this is unclear, refer again to Figure 6-1. Change the word "read" in the figure to "write".

Adjusting Network Buffer Sizes

The network buffer size is the maximum amount of data that can be sent from or received by a machine in a single packet. You can improve network performance by properly configuring buffer sizes.

There are two important things to keep in mind when adjusting buffer sizes. First, large buffers use more memory. If you specify too large a buffer size, other processes may suffer for the lack of available memory. Second, when two machines with different buffer sizes are involved in a transaction, the smaller of the two buffer sizes is used. This is called the "effective buffer size."

The following sections provide some tips on configuring buffer sizes for maximum performance.

Checking the Buffer Size

To find the size of the network buffer that is being used for LAN Manager Client communications between two computers, enter the **net stat** command on one of the two computers. The *bufsz* column of the **net stat** output lists the effective buffer size for every machine to which the computer is connected.

Buffer Size Is Automatically Configured by Default

These two parameters in the file */usr/lib/xnet/xnetrc* specify the size and number of the network buffers:

Name	Configures
NBIOSIZ	buffer size
NBUFALLOC	number of buffers

If these parameters are set to zero, LAN Manager Client checks the amount of available memory and sets NBIOSIZ and NBUFALLOC to what it thinks are good values. To override the autoconfigured default, set the parameter to any value other than zero.

For more information on the *xnetrc* file, see "Network Parameter Descriptions" earlier in this chapter.

Calculating the Maximum Buffer Size

The maximum allowable size of the network buffer (NBIOSIZ) is determined by the size of the largest streams buffer configured on your machine. Use this formula to calculate the maximum size of the network buffer:

Max Network Buffer size = (Max Streams Buffer Size * 2) – 48 bytes

To find out what the maximum streams buffer is on your machine, check these parameters:

nblk8172	N8K	0
nblk4096	N4K	0
nblk2048	N2K	10
nblk1024	N1K	20
nblk512	N512	10
nblk256	N256	10
nblk128	N128	10
nblk64	N64	30
nblk16	N16	60
nblk4	N4	128

These parameters are found in the file */etc/conf/cf.d/mtune*.

These parameters set the number of streams buffers of various sizes, from 4 bytes to 8 Kbytes. In the previous example, there are no 8- or 4-Kbyte buffers configured, there are ten 2-Kbyte buffers, thirty 64-byte buffers, and so on. Because no 8- or 4-Kbyte buffers are configured, the maximum streams buffer size configured in the example is 2 Kbytes, or 2048 bytes. Thus, the maximum network buffer size is 4048 bytes. Rounding down to the nearest 1-Kbyte increment, this gives us a maximum network buffer size of 3 Kbytes. By configuring some 4-Kbyte streams buffers, you can raise this value to 7 Kbytes.

For more information on the streams buffer parameters, see "Network Parameter Descriptions" earlier in this chapter.

Chapter 7

Building a Remote Network with UUCP

This chapter explains how to use the ODT-NET UUCP package to build a remote network system for your computer using a normal telephone line and a modem.

NOTE: UUCP is not a terminal emulation program. If you want to use your modem to dial into another computer and log on, you should refer to *Administering ODT-OS* in the *Administrator's Guide* and follow the instructions for adding dial-in and dial-out modems. If you plan to do extensive file transfers between UNIX and XENIX systems, you should set up a UUCP connection.

What Is UUCP?

The UUCP package permits UNIX and XENIX systems to communicate as part of a remote network. The name UUCP is an acronym for "UNIX-to-UNIX Copy." The UUCP package consists of a group of programs that provide the following capabilities:

- Remote file transfer (**uucp**)
- Remote command execution (**uux**)
- Mail to and from remote sites (via **mail**)

Used primarily over phone lines, UUCP can be used to connect with specific remote machines on a demand or scheduled basis, and by either dialing out or allowing other machines to call in.

UUCP uses a batch method to manage communications traffic, storing (or "spooling") requests for later execution when actual contact is made between systems. When UUCP commands are executed, work files and any data files needed are created in */usr/spool/uucp*. The program **uucico** scans this directory for the instructions contained in any work files and executes them. Although it is possible to execute commands immediately, most systems call other systems according to a daily schedule (usually during the evenings to reduce connection costs).

How to Use This Chapter

This chapter describes how to build a UUCP system and covers both hardware installation and software configuration. There are also sections on routine maintenance and troubleshooting.

The following is an outline of what must be done to set up your UUCP network:

1. Connect and configure a modem or direct wire.
2. Configure the UUCP software using **uuinstall**.
3. Create login accounts for any sites that will be calling your system.
4. Test your connections with each remote site.

The most important task of configuring UUCP is the editing of several control files that act as the database for UUCP. The next few sections describe the function of these files, and "Configuring UUCP on Your System" explains the information that these files contain. The **uuinstall** utility edits these files for you and explains each entry. **uuinstall** also includes an extensive help facility. Read "Configuring UUCP on Your System" carefully before running **uuinstall** to understand the UUCP database.

What You Need

To set up your UUCP communication system, you need:

- At least one RS-232 serial line (or serial port) on your computer to use for UUCP.
- The UUCP and MAIL packages extracted from your UNIX System distribution using **custom**(ADM).
- A modem. Supported modems include models by Hayes, Penril, Ventel, Vadic, Rixon, AT&T, and Telebit. You can supply **Dialers** entries or dialer programs for other modems. (For best results, use dialer programs.) Instructions for the Hayes Smartmodem 1200 or 2400 and compatibles are given later in this chapter.
- A standard telephone jack for access to the telephone system.
- A cable to connect the serial port to the modem.

UUCP Commands

UUCP programs are divided into two categories: user programs and administrative programs. The paragraphs that follow describe the programs in each category.

User Programs

The user programs for basic networking are in */usr/bin*. No special permission is needed to use these programs. These commands are all described in *Using ODT-NET* in the User's Guide.

Administrative Programs

Most of the administrative programs, control files, and scripts are in */usr/lib/uucp*. Two exceptions are **uuinstall** and **uulog**, which are in */etc* and */usr/bin*, respectively.

uulog Displays the contents of a specified computer's log files. Log files are created for each remote computer your computer communicates with. The log files contain records of each use of **uucp**, **uuto**, and **uux**.

uuclean Cleans up the spool directory. It is normally executed from a shell script called **uudemon.clean**, which can be set up to be run by **cron**.

uutry Tests call-processing capabilities and does a moderate amount of debugging. It invokes the **uucico** daemon to establish the communications link.

uucheck Checks for the presence of basic networking directories, programs, and support files. It also checks the **Permissions**, **Systems**, and **Devices** files for syntax errors.

uuinstall Configuration script for UUCP control files and ports.

UUCP Directories

There are three directories associated with UUCP:

/usr/spool/uucp	The working directory for UUCP. Work files, log files, and all UUCP communications traffic are stored here.
/usr/spool/uucppublic	This is the publically readable or writable target directory used for most file transfers.
/usr/lib/uucp	Most of the UUCP programs are stored here, as well as the supporting database/control files. The main user programs, including **uux** and **uucp**, are found in */usr/bin*.

/usr/lib/uucp also contains configuration files for UUCP (distinguished by their capitalized names). The most important to understand are:

Systems	Contains information needed to establish a link to a remote computer, including the name of the connecting device associated with the remote computer, when the computer can be reached, telephone number, login sequence, and password.
Permissions	Defines the access level granted to computers when they attempt to transfer files or remotely execute commands on your computer.
Devices	Contains information concerning the port name, speed, and type of the automatic call units (modems), direct links, and network devices.

UUCP Background Programs

uucp traffic is managed by three *daemons*, or supervisory programs, that run in the background, handling file transfers and command executions. (The daemons can also be executed manually as commands.)

uucico	Selects the device used for the link, establishes the link to the remote computer, performs the required login sequence and permission checks, transfers data and executes files, logs

results, and (if requested) notifies the user by **mail** of transfer completions. When the local **uucico** daemon calls a remote computer, it "talks" to the **uucico** daemon on the remote computer during the session.

uuxqt Executes remote program execution. It searches the spool directory for execute files (X.*file*) that were sent from a remote computer. When an X.*file* file is found, **uuxqt** opens it to get the list of data files that are required for the execution. It then checks to see if the required data files are available and accessible. **uuxqt** also verifies that it has permission to execute the requested command.

uusched Schedules the queued work in the spool directory. Before starting the **uucico** daemon, **uusched** randomizes the order in which remote computers are called.

How UUCP Works

When you enter a UUCP command, the program creates a work file and usually a data file for the requested transfer. The work file contains information required for transferring the file(s). The data file is a copy of the specified source file. After these files are created in the spool directory, the **uucico** daemon is started.

The **uucico** daemon attempts to establish a connection to the remote computer. First it gathers the information required for establishing a link to the remote computer from the **Systems** file. This is how **uucico** knows what type of device to use in establishing the link. Next, **uucico** searches the **Devices** file looking for the devices that match the requirements listed in the **Systems** file. After **uucico** finds an available device, it attempts to establish the link and log in on the remote computer.

When **uucico** logs in on the remote computer, it starts the **uucico** daemon on the remote computer. The two **uucico** daemons then negotiate the line protocol to be used in the file transfer(s). The local **uucico** daemon then transfers the file(s) that you are sending to the remote computer. The remote **uucico** places the file in the specified pathname(s) on the remote computer. After your local computer completes the transfer(s), the remote computer may send files that are queued for your local computer. The remote computer can be denied permission to transfer these files with an entry in the **Permissions** file. (This is also affected by directory permissions.) If this is done, the remote computer must establish a link to your local computer to perform the transfers. A remote computer can also request files.

If the remote computer or the device selected to make the connection to the remote computer is unavailable, the request remains queued in the spool directory. If set up to run by **cron**, each hour (default), **uudemon.hour** starts the **uusched** daemon. When the **uusched** daemon starts, it searches the spool directory for the remaining work files, generates the random order in which these requests are to be processed, and then starts the transfer process (**uucico**) described in the previous paragraphs.

A Sample UUCP Transaction

The following traces the execution of a **uucp** command:

1. A user on a system called "kilgore" wishes to send a copy of the file *minutes.01.10* to a remote system called "obie". To accomplish this, the user enters the following command:

   uucp minutes.01.10 obie\!usr/spool/uucppublic

   Note that the exclamation point need only be escaped (preceded by a "\") if the **csh** is used; the Bourne shell (**sh**) does not require this.

2. A work file is created in the */usr/spool/uucp/obie* directory, C.obie*xxxx*, where *xxxx* is the job number.

3. The **uusched** daemon schedules the request for execution by **uucico**.

4. When the execution time is reached, **uucico** first checks the **Systems** file and confirms that "obie" is a recognized system and that a call is permitted at this time.

5. Using the information in the **Systems** file, **uucico** next locates the modem device and tty port associated with it as stored in the **Devices** file.

6. Using the phone number in the **Systems** file and the modem type from the **Devices** file, **uucico** uses the appropriate modem commands from the **Dialers** file (or runs a dialer program from the */usr/lib/uucp* directory) to connect to the remote system.

UUCP Control Files (sites: *kilgore* and *obie*)	
Systems:	`obie Any ACU 2400 14081234567 \` `  --ogin:-BREAK-ogin: nuucp ssword: mavra`
Devices:	`ACU tty1A - 2400 dialHA24`

Permissions:	`LOGNAME= ukilgore MACHINE= kilgore \`
	`  READ=/usr/spool/uucppublic:/usr/kilgore \`
	`  WRITE=/usr/spool/uucppublic:/usr/kilgore \`
	`  REQUEST=no  SENDFILES=call \`
	`  COMMANDS=rmail:rnews:uucp`

7. **uucico** creates a lock file (LCK..tty*xx*) to lock the serial line, and a lock file (LCK..obie) to lock the called system in the directory */usr/spool/uucp*.

8. **uucico** uses the login sequence and password defined in the **Systems** file to log-in to "obie", whose own **uucico** confirms that "kilgore" is recognized before beginning the actual transaction.

9. The calling system, "kilgore," (sometimes known as the *guest*) is said to be the "master" of the transaction; the called system, "obie," (also known as the *host*) is said be the "slave." The slave **uucico** checks the local **Permissions** file to confirm that the master is authorized to transfer the file.

10. The master ("kilgore") transmits the file in packets that are checked for errors and retransmitted if garbled. During reception, the file is stored in a temporary file (TM.*xxxx*) in the */usr/spool/uucp* directory. When the transfer is complete, the file is moved to the proper destination, in this case */usr/spool/uucppublic/minutes.01.10*.

11. Each machine records its side of the transaction in log files. For example, "obie" would have the exchange recorded in a file called */usr/spool/uucp/.Log/uucp/kilgore*.

12. Unless the slave system "obie" has requests of its own, a hangup request is sent, the connection is terminated, and the lock files removed.

For remote command execution (via **uux**), an execute X.*file* is created in the */usr/spool/uucp* directory. The **uuxqt** daemon scans this directory for work, checks the **Permissions** file to confirm permission to execute the command, then executes it. The device name should have the form

/dev/tty*nn*

where *nn* is the number of the corresponding line. For example, */dev/tty1a* usually corresponds to COM1. You need the name of the actual line for later steps.

The serial port should be owned by **uucp**. To make sure the line is owned by **uucp** enter this command:

chown uucp /dev/tty*nn*

where *nn* is the number of the corresponding line.

Connect a Serial Cable

You connect two computers together using an RS-232 cable. The actual pin configurations sometimes vary between machines.

Typically, the cable should connect pins 2, 3, and 7 on one computer to the same pins on the second computer. Sometimes the cable must be *nulled*, which means that pin 2 on one machine is connected to pin 3 on the other, and vice versa. Since the connections can vary, check the hardware manuals for each computer to determine the proper pin connections.

Testing a Connection

For this section, *tty2a* is used as the example serial port for both machines.

To test the wire connection between two machines:

1. Disable the serial lines on each machine. On each computer, enter the command:

   disable /dev/tty2a

   Be sure to disable the modem control line as well:

   disable /dev/tty2A

2. Attach one end of the serial wire to one of the machines. Attach the other end to the standard data port of a terminal.

3. Enter this command at the computer:

   (stty 9600; date) < /dev/tty2a > /dev/tty2a

   tty2a is our example serial line, and the date command provides sample output.

   You should see the output of the **date** command appear on the terminal screen. Repeat this procedure on the other machine.

   If this doesn't work, check the following:

   - The wire is plugged in properly at each end.
   - The continuity of the wire.
   - The terminal is configured correctly (baud rate, parity, etc.).
   - The serial line is disabled.
   - You are using the correct pin numbers.

NOTE: An unterminated serial cable can cause serious system problems. Do not leave serial cable dangling.

Connecting Remote UUCP Systems with a Modem

With a modem, you can communicate with computers over standard phone lines. These are the steps to install a modem:

- Choose a serial port.
- Set the dialing configuration.
- Connect the modem and set the switches or registers.
- Test the connection.

The following sections explain each step in detail. Make certain you are aware of special services on your phone line; "call waiting" can disrupt UUCP communications.

Choose a Serial Port

Choose the RS-232 serial port you want to use with the system and connect to the modem. If there are no lines available, you must install a new serial line or make one available by removing any device connected to it. If you remove a terminal, make sure no one is logged in.

Find the name of the device special file associated with the port by referring to *Administering ODT-OS*. The device name should have the form

/dev/tty*nn*

where *nn* is the number of the corresponding port. For example, */dev/tty1A* corresponds to COM1. You need the name of the actual port for later steps.

NOTE: */dev/tty1a* and */dev/tty1A* are the same physical port; **tty1a** is used for terminals and direct connections; *tty1A* is used for modem connections.

The serial port should be owned by *uucp*. To make sure the line is owned by *uucp* enter this command:

chown uucp /dev/tty*nn*

where *nn* is the number of the corresponding line.

Set the Dialing Configuration

The modem can be used to both send and receive calls. You must set the appropriate switches on the modem. The instructions that follow are specific to Hayes-compatible modems, but other modems are supported. You should refer to the modem manual for connection instructions and see the section "Adding Dial-Out Entries to the Devices File" under "Configuring UUCP on Your System" for a complete list of supported modems and dialer programs. (If you are setting up a Hayes Smartmodem 2400 or compatible, see the next section for configuration instructions.) Follow these steps to configure a Hayes Smartmodem 1200 or compatible modem:

1. Remove the front cover of the modem and locate the 8-position configuration switch. (See the modem reference manual for instructions on how to locate the switch on your particular model.)

2. Set the switches as they appear here:

	1	2	3	4	5	6	7	8
up	●	●		●	●	●	●	
down			●					●

Table 7.1 explains each of the settings.

3. Replace the front cover.

Table 7.1.

Hayes-Compatible Switch Settings

Switch	Position	Function
1	up*	Modem responds to DTR from computer
	down	Modem forces DTR high, so no signal is required from computer
2	up*	Result codes in English
	down	Numeric result codes
3	up	No result codes
	down*	Result codes are sent in response to each modem command
4	up*	Commands are echoed
	down	Commands are not echoed
5	up*	Modem will answer phone
	down	Modem will not answer phone
6	up*	CD asserted when carrier is actually present
	down	CD and DSR forced high
7	up*	Modem attached to single-line phone
	down	Modem attached to multi-line phone
8	up	Modem does not recognize dialing commands
	down*	Modem recognizes dialing commands

If you have a different modem, consult your reference manual for the proper switch settings to both send and receive calls.

Connect the Modem

Once your modem's dialing configuration is set, you are ready to connect the modem to your computer. For proper modem operation, the RS-232 cable must provide the pin connections in Table 7.2.

Note that the computer's serial connector must have a DTE (Data Terminal Equipment) configuration. The modem is assumed to have a DCE (Data Communications Equipment) configuration. If both pieces of equipment have DTE or DCE, you need a null modem connection.

Table 7.2.

Pin Connections

Name	Computer (DTE)	Modem (DCE)
Protective Ground	1	1
Transmit Data (TD)	2	2
Receive Data (RD)	3	3
Request to Send (RTS)	4	4
Clear to Send (RTS)	5	5
Data Set Ready (DSR)	6	6
Signal Ground (SG)	7	7
Carrier Detect (CD)	8	8
Data Terminal Ready (DTR)	20	20

These connections are explained in the reference manual for your modem.

Review the installation instructions given in your modem's manual, then follow these steps:

1. Connect the RS-232 serial cable to the serial-line connector on the modem, then to the serial-line connector on your computer. Make sure the cable is fully connected. (A 2-3-7 pin terminal cable is not sufficient. We suggest a ribbon cable to connect all appropriate wires.)

2. Plug the telephone line cable into the "line" or "wall" connector on the modem, then into the wall jack.

3. Plug in the power cord of the modem.

Configuring a Hayes 2400 or Compatible Modem

Although most aspects of modem installation are similar, a Hayes 2400 Smartmodem or compatible modem requires online configuration if it is to be used as a dial-in line. Note that the Hayes 2400 does not answer the phone with a 2400 baud carrier if it was not setup with 2400 baud commands.

1. Make sure that the **Devices** file contains an entry for the line:

   ```
   ACU  ttynn  -  300-2400  /usr/lib/uucp/dialHA24
   ```

2. Disable **getty** temporarily with:

   disable tty*nn*

3. You must then configure the modem by issuing set up commands via **cu**(C). Enter:

   cu -s2400 -l/dev/tty*nn* **dir**

   where *nn* is the "tty" number of the serial line. Press **Return**.

4. Next, enter the following commands to configure the modem. They are saved in the modem's non-volatile memory. If you do not want to save the settings, do not enter the last command (AT&W). Commands are in the left column and short descriptions of what they do are in the right column. Follow each command with a **Return**:

AT&F	Fetch factory configuration.
ATT	Tone dialing.
ATL0	Low speaker volume.
AT&D2	Set DTR "2": go on hook when DTR drops.
AT&C1	Set dcd "1": dcd tracks remote carrier.
ATS0=1	Answer phone after 1 ring (AA light should come on).
ATS2=128	Disable modem escape sequence.
ATE0	No echo (modem will no longer echo what is sent to it).
ATQ1	Quiet mode (modem will not respond with "OK" after this command or any that follow).
AT&W	Saves settings in non-volatile memory.

5. Exit from **cu** by entering a "tilde" and a "period", followed by **Return**:

   ~.

   (Sometimes it is necessary to press **Return** once before entering the tilde-period.)

6. Re-enable the port only if you wish the modem to receive calls:

   enable tty*nn*

The modem is now configured and ready for testing.

Variable Rate Modems

Some modems can determine the connection baud rate from the carrier sent by a remote system. These modems inform the local system of the connection baud rate before issuing the carrier detect signal. The Hayes 2400 dialer supplied with UUCP detects different connection baud rates and informs UUCP and **cu** when it exits with a successful connection.

The speed fields in **Devices** and **Systems** can specify a range of baud rates for a connection. If a dialer supports baud rates from 300 to 2400 baud, enter the baud rate range in the speed field of **Devices** as follows:

300-2400

If a dialer or modem does not allow variable baud rates, place a single baud in the speed field. If a remote system supports several different speeds, place the range of baud rates in the speed field of **Systems**. If the remote system connects at a single baud rate, place that number in **Systems**. UUCP passes the intersection of the **Systems** and **Devices** baud rate ranges to the dialer when connecting. If the dialer connects outside of the baud range, it returns a bad baud rate error. Otherwise, it returns the baud rate of the connection.

Test the Modem

As the last step of the modem installation, you should test the modem to make sure that it can send and receive calls. Once you have verified that the modem is working, you can begin to use the communications system.

To test the modem, follow these steps:

1. If you are using a Hayes 1200 or compatible, make sure the volume switch on the modem is at an appropriate level. You must be able to hear the modem to carry out this test successfully. Refer to your modem reference manual for the location of this switch.

2. Ensure that the **Systems** file has an entry for the system you intend to call, and that the **Devices** file has a matching entry for tty*nn*.

3. Start the **uutry** program by entering:

   /usr/lib/uucp/uutry -x6 *sitename*

4. Listen carefully to the modem. You should hear each digit as the number is dialed, then hear a high-pitched signal when the other modem connects, followed by silence.

5. The dialer automatically disconnects any call that it cannot complete. Do not interrupt using **Del** or otherwise stop **uutry**. Let the dialer hang up.

6. If the signal is not present, make certain:

   - you have connected the modem to the telephone jack
   - the jack is connected to the phone system
   - you gave the correct phone number in the **Systems** file

7. If you do not hear the modem dial, make certain:

   - the volume switch is up
   - the modem is connected to the correct serial line and that the cable connection is tight
   - you gave the correct tty line in the **Devices** file
   - modem's power is on
   - there are no LCK.. files in */usr/spool/uucp*.

8. **uucico** only allows one call to a given system every ten minutes. You can wait before retrying, or remove the file associated with the site you are calling in the directory */usr/spool/uucp/Status*.

Configuring UUCP on Your System

To configure your UUCP system, you must edit a series of files that contain information about, and control the actions of the UUCP programs. The UUCP control files are in the */usr/lib/uucp* directory. You can modify these files with a standard text editor, or use the **uuinstall**(ADM) program as described here. The descriptions in the latter part of this section provide details on the structure of these files so you can edit them manually.

An Important Consideration: Call or Be Called?

There are three ways to configure a UUCP site:

- a *dial-in* only site.
- a *dial-out* only site.
- a dial-in/out site.

As a dial-in site, other computers call up and log in to your system. They can transfer files and execute certain commands.

As a dial-out site, your computer calls up other computers and logs in. Your computer initiates file transfers to and from the remote machine, as well as local and remote command execution.

Setting Up the Control Files with uuinstall

The rest of this section is concerned with the configuration or control files that act as the UUCP database. The **uuinstall**(ADM) utility provides a simple way to configure these files. Read the rest of the chapter to familiarize yourself with the descriptions of each file and the entries required. The **uuinstall** utility includes a complete series of help files (accessed by pressing **?** while in the menus) so it is not necessary to keep referring to the documentation. When you have some understanding of how each of the control files is used, follow this procedure:

1. Invoke **uuinstall** by logging in as **root** and entering the following command:

   /etc/uuinstall

Δ **sysadmsh** users select: System→Configure→Network→UUCP

The main **uuinstall** menu is displayed:

```
          UUCP Administration Utility
          ===========================

1. Display or update site name
2. Display or update list of remote sites       (Systems)
3. Display or update direct- or dial-out lines  (Devices)
4. Display or update direct- or dial-in lines
5. Check consistency of UUCP files
6. Test connection with remote site
7. Convert old UUCP files to new format

Choose an option (1-7), or enter "q" to quit :
```

Use the **uuinstall** options as follows:

- If you did not set your site name at installation time, or you wish to change your site name, do so using the first option.
- Choose the devices to be used for dialing-in or out and enter them in the **Devices** file using the "Display or update dial-in or dial-out devices" option.
- Identify sites your system will have contact with by creating entries in the **Systems** file with the "Display or update list of remote sites" option.
- Add the tty lines to be used to the */etc/inittab* file using the "Display or update line connections" option.

NOTE: If you wish any changes made to */etc/inittab* to be permanent, you must also make the same change to */etc/conf/cf.d/init.base*. This is because each time the kernel is relinked (as when a driver is added or tunable parameter changed), */etc/inittab* is reconstructed from the entries found in */etc/conf/cf.d/init.base*.

2. If other systems will be calling yours, create login accounts as described in "Creating Login Accounts for Sites Dialing-In" later in this section.
3. If other systems will be calling yours, define a security scheme that includes what commands and directories can be used in the **Permissions** file.

When you install the UUCP system, or make any modifications, you should be logged in as super user (root). Virtually all of the UUCP files are writable only by the super user, and many of them are also readable and executable only by **root** and **uucp**. Make sure when you are done that all of the UUCP files are owned by **uucp** and not **root**. UUCP does not work correctly if it cannot read or execute its files. To check the permissions of the UUCP files, use the following commands:

```
cd /
fixperm -n -v -dUUCP /etc/perms/*
```

This command displays any UUCP files with incorrect permissions.

NOTE: The files **Systems** and **Permissions** contain unencrypted passwords, and they should, therefore, be readable only by **uucp** (and **root**). Note also that the program */usr/bin/ct* must be owned by **root** and not by **uucp** to work correctly.

Changing your Site Name

Use the **uuinstall** utility to change the name of your UUCP site. If you wish to change your sitename manually, or wish to maintain different names on different networks, refer to the *Administering ODT-OS* in the *Administrator's Guide*.

Selecting and Defining a UUCP Port

As discussed earlier, you must select a serial port, disable it if it is to be used for dial-out only, or enable it for dial-in, and edit the serial line entry in the */etc/inittab* file.

NOTE: If you wish any changes made to */etc/inittab* to be permanent, you must also make the same change to */etc/conf/cf.d/init.base*. This is because each time the kernel is relinked (as when a driver is added or tunable parameter changed) */etc/inittab* is reconstructed from the entries found in */etc/conf/cf.d/init.base*.

1. Select the serial line. Use a line with modem control (for example, */dev/tty1A*) for a dial-in or dial-out line, or a line without modem control (for example, */dev/tty2a*) for a direct connection. For more information, see "Choose a Serial Port" earlier in this section.

2. Disable the serial line. If you are using a modem, be sure it is installed and tested. If the serial line is to be a dial-in line, substitute **enable** for disable and enter the command:

   disable /dev/tty*nn*

   where *nn* is the number of your serial line. If the line is already disabled/enabled, the command displays an error message that you can safely ignore.

3. Edit the */etc/inittab* file. This file contains a list of possible login terminals. Enter the following command to display the current entries for the different serial lines:

   cat /etc/inittab

   tty entries have the following form:

   t*n*:2:respawn:/etc/getty tty*n* m

   where *n* is the **tty** number. If you need to change an entry, you can do so with a text editor. For more information on the */etc/inittab* file and the various control codes, see the **getty**(M) and **inittab**(F) manual pages.

   For example, an entry for a dial-out line (connected to a modem) might

   look like this:

   ```
   t2A:2:respawn:/etc/getty tty2A m
   ```

   An example entry for a direct line between two computers might be:

   ```
   t2a:2:respawn:/etc/getty tty2a m
   ```

   If the line is to be shared between dial-in and dial-out, ensure that it has an appropriate entry in */usr/lib/uucp/Devices* and in */etc/inittab*.

Creating Login Accounts for Dial-in Sites

A dial-in site must provide a login entry for the sites that call it. These entries are placed in the */etc/passwd* file.

A UUCP login entry has the same form as an ordinary user login entry but it has a special login directory and login program instead of the normal user directory and shell. (Refer to *Administering ODT-OS* in the Administrator's Guide for more information on creating login accounts)

NOTE: "uucp" should not be used as the name of a UUCP user or login account; it is the name of the uucp owner/administrator.

To create a UUCP login entry, follow these steps:

1. Choose a new user name and a user ID (identification number) for the UUCP login. The name can be any combination of letters and digits that is no more than eight characters long. The user ID must be an integer in the range 50 to 65535.

   Make sure the name and ID are unique. A UUCP login entry must not have the same name or ID as any other login entry.

2. To create the new account, invoke the **sysadmsh** and make the following selection:

   Accounts→User→Create

3. Use the following information to create the account:

   Login shell: /usr/lib/uucp/uucico
   Home directory: /usr/spool/uucppublic

   Passwords are optional, but recommended, for UUCP logins.

Adding Entries for Remote Sites to the Systems File

The **Systems** file (*/usr/lib/uucp/Systems*) contains the information needed by the **uucico** daemon to establish a communications link to a remote computer. Each entry in the file represents a computer that can be called by your computer.

NOTE: After creating the **Systems** file, and each time you modify it, you must log in as user **mmdf** and execute the following commands:

```
cd /usr/mmdf/table
tools/uulist
dbmbuild
```

This ensures that the MMDF routing mechanism properly handles traffic for the new or modified sites.

In addition, the **Systems** file can be configured to prevent any computer that does not appear in this file from logging in on your computer. More than one entry may be present for a particular computer. The additional entries represent alternative communication paths that will be tried in sequential order.

NOTE: If you are setting up your system as a *dial-in* only (passive) site that never initiates calls, you only need to add the names of the systems that will be calling you.

Each entry in the **Systems** file has the following format (each field must be separated by a space):

sitename schedule device speed phone login-script

where:

sitename	Contains the node name of the remote computer.
schedule	Is a string that indicates the day-of-week and time-of-day when the remote computer can be called.
device	Is the device type that should be used to establish the communications link to the remote computer.
speed	Indicates the transfer speed of the device used in establishing the communications link.

phone	Provides the phone number of the remote computer for automatic dialers.
login-script	contains login information (also known as a "chat script").

The Schedule Field

The *schedule* consists of three subfields. The first, *day*, is required. The other two, *time* and *retry*, are optional. The syntax is as follows:

day[*time*][*;retry*]

The *day* subfield can contain the following keywords:

Su Mo Tu We Th Fr Sa	For individual days.
Wk	For any weekday (Mo Tu We Th Fr).
Any	For any day.
Never	For a passive arrangement with the remote computer. If the *schedule* field is **Never**, your computer never initiates a call to the remote computer. The call must be initiated by the remote computer. In other words, your computer is in a passive mode in respect to the remote computer (see discussion of **Permissions** file).

The optional *time* subfield should be a range of times in 24-hour clock format, such as 0800-1230. If no *time* is specified, any time of day is assumed to be allowed for the call. A time range that spans 0000 is permitted. For example, **0800-0600** means all times are allowed other than times between 6 am and 8 am.

For example, the following permits calls on Mondays, Wednesdays, and Fridays between the hours of 9 am and noon (the *schedule* field is in boldface for clarity):

```
grebe MoWeFr0900-1200 ACU D1200 14087672676 \
    ogin: nuucp ssword: Crested
```

You can also specify more than one set of *day* and *time* entries by separating them with commas. This is useful for more complex specifications. The following example allows calls from 5:00 pm to 8:00 am, Monday through Thursday, and calls any time Saturday and Sunday.

The following example would be an effective way to call only when phone rates are low, if immediate transfer is not critical:

```
gorgon Wk1700-0800,SaSu ACU D1200 14087672676 \
    ogin: nuucp ssword: DontLook
```

The optional subfield, *retry*, is available to specify the minimum time (in minutes) before a retry, following a failed attempt. The subfield separator is a semicolon (;). For example, the following is interpreted as "call any time, but wait at least 9 minutes before retrying after a failure occurs":

```
Any;9
```

NOTE: By default, UUCP uses an "exponential backoff" method to retry failed calls. After the initial failure, a second call is made in 5 minutes. This interval expands as the number of unsuccessful attempts increases. The retry field is used to override the default.

The Device Field

The *device* field selects the device type, in most cases an ACU (Automatic Calling Unit). For example, the keyword used in the following field is matched against the first field of **Devices** file entries:

Systems:
```
gorgon Any ACU D1200 14087672676 \
  ogin: nuucp ssword: DontLook
```

Devices:
```
ACU tty2A - D1200 hayes
```

The Speed Field

This field can contain a letter and speed (for example, C1200, D1200) to differentiate between classes of dialers (refer to the discussion on the **Devices** file, *speed* field). Some devices can be used at any speed, so the keyword **Any** can be used. However, we recommend that you specify the

actual range of speeds that can be used. (If **Any** is used in both **Systems** and **Devices** entries, 1200 is assumed.) For example, this field must match the *speed* field in the associated **Devices** file entry:

```
Systems:   gorgon Any ACU D2400-9600 14087672676 \
             ogin: nuucp ssword: DontLook

Devices:   ACU ttyla - D2400-9600 hayes2400
```

If information is not required for this field, use a hyphen (-) as a place holder for the field.

The Phone Field

This field is used to provide the phone number used for the modem dialer. The phone number is made up of an optional alphabetic abbreviation and a numeric part. If an abbreviation is used, it must be one that is listed in the **Dialcodes** file. For example:

```
Systems:    gorgon Any ACU D1200 CA2676 \
              ogin: nuucp ssword: DontLook

Dialcodes:  CA 9=408767
```

In this string, an equal sign (=) tells the ACU to wait for a secondary dial tone before dialing the remaining digits. A dash in the string (-) instructs the ACU to pause 2 seconds before dialing the next digit.

If your computer is connected to a LAN switch or port selector, you can access other computers that are connected to that switch. The **Systems** file entries for these computers will not have a phone number in the *phone* field. Instead, this field contains the token that must be passed on to the switch so it knows which computer your computer wishes to communicate with. (This is usually just the system name.) The associated **Devices** file entry should have a **\D** at the end of the entry to prevent translation using the **Dialcodes** entry.

The Login-Script Field

The login-script is used to open communications between modems, plus recognize and send proper login and password sequences. The script is given as a series of space-separated fields and subfields of the following format:

expect send

where *expect* is the string that is received, and *send* is the string that is sent when the *expect* string is received.

The *expect* field can be made up of subfields of the following form:

expect[*-subsend-subexpect*]...

where the *subsend* is sent if the prior *expect* is not successfully read and the *subexpect* following the *subsend* is the next expected string. To make this distinction clear: the send-expect sequence sends a string if the expect string is received, the subsend-subexpect sends only if the prior expect string is not received within 10 seconds.

For example, with "login--login", the UUCP program expects "login". If a "login" is received, it goes on to the next field. If it does not get "login", it sends nothing followed by a carriage return, then looks for "login" again. If no characters are initially expected from the remote computer, the characters "" (null string) should be used in the first *expect* field. Note that all *send* fields are sent followed by a carriage return unless the *send* string is terminated with a \c.

If an *expect* string starts with a dash, it is interpreted as a null *expect* string followed by a *subsend* string. For example, "--login:" sends a carriage return and then expects a "login:".

The *expect* string need not be complete; only the trailing characters must be specified, as in "ogin:". This avoids difficulties with login strings that use an uppercase letter as in "Login:" or "Password:", and also difficulties when the line is shared by dial-in and dial-out.

Creating Login Scripts

This section explains in greater detail how to create a login (chat) script.

Consider the following sample **Systems** file entry:

```
terps Any ACU 1200 18005211980 "" \r ogin:-BREAK-ogin: \
  uucpx word: ichore
```

This is how this script would work during connection:

1. Nothing is expected initially.
2. A carriage return is sent and the script waits for the prompt "ogin:" (login:).
3. If it does not receive "ogin:", send a BREAK signal.
4. When "ogin:" is finally received, send the login name **uucpx**.
5. When the prompt "word:" (for Password:) is received, send the password "ichore".

Login (chat) scripts often require some experimentation. There are cases that require one or more BREAK sequences before presenting a login (this is often true with variable speed modems). If you cannot obtain the necessary login sequence from the system administrator for a given site, it is a good idea to connect with the site manually. You can accomplish this using **cu** and find out what must be sent to generate a login prompt. (You can also connect with a system using a **uutry** for debugging; see "Debug Transmissions" under "Troubleshooting" for details.) There are several escape characters that cause specific actions when sent during the login sequence, some of which correspond to keystrokes; these should be included in the script where necessary. See Table 7.3.

Table 7.3.

Login (Chat) Script Escape Sequences

Character	Description
\N	Sends a null character (ASCII NUL).
\b	Sends or expects a backspace character.
\c	If at the end of a string, suppresses the carriage return that is normally sent. Ignored otherwise.
\d	Delays two seconds before sending or reading more characters.
\p	Pauses for approximately ¼ to ½ second.
\E	Starts echo checking. (From this point on, whenever a character is transmitted, it waits for the character to be received before doing anything else.)
\e	Turns echo check off.
\n	Sends or expects a new-line character.
\r	Sends or expects a carriage-return.
\s	Sends or expects a space character.
\t	Sends or expects a tab character.
\\	Sends or expects a \character.
EOT	Sends EOT (end of transmission or **Ctrl**-d)
BREAK	Sends a BREAK signal.
\K	Same as BREAK.
\ddd	Collapses the octal digits (ddd) into a single character.
""	Expects a null string.

Limiting Access with the Permissions File

If other machines will be dialing into your system, the **Permissions** file (*/usr/lib/uucp/Permissions*) specifies the permissions that remote computers have with respect to login, file access, and command execution. There are options that restrict the remote computer's ability to request files and its ability to receive files queued by the local site. Other options specify the commands that a remote site can execute on the local computer.

Structuring Permissions File Entries

Each entry is a logical line with physical lines terminated by a \ to indicate continuation. Entries are made up of options delimited by spaces. Each option is a name-value pair in the following format:

name=value

Note that no spaces are allowed within an option assignment.

Comment lines begin with a crosshatch sign (#) and they occupy the entire line up to a newline character. Blank lines are ignored (even within multi-line entries).

There are two types of **Permissions** file entries:

LOGNAME	Specifies the permissions that take effect when a remote computer calls your computer.
MACHINE	Specifies permissions that take effect when your computer calls a remote computer.

Permissions File Restrictions

When using the **Permissions** file to restrict the level of access granted to remote computers:

- All login IDs used by remote computers to log in for UUCP communications must appear in only one LOGNAME entry.
- Any site that is called whose name does not appear in a MACHINE entry, has the following default permissions/restrictions:
  - — Only local send and receive requests are executed.
  - — The remote computer can send files to your computer's */usr/spool/uucppublic* directory.
  - — The commands sent by the remote computer for execution on your computer must be one of the default commands, usually **rmail**.

NOTE: When a remote machine calls you, unless you have a unique login and password for that machine, you do not know if the machine is who it claims to be.

Permissions Options

This section lists some of the available options. See the examples at the end of this chapter and the **permissions**(F) manual page for details.

REQUEST
Specifies whether the remote computer can request to set up file transfers from your computer.

SENDFILES
Specifies whether your computer can send the work queued for the remote computer. When a remote computer calls your computer and completes its work, it may attempt to take work your computer has queued for it.

READ and **WRITE**
Specify the various parts of the file system that **uucico** can read from or write to. The READ and WRITE options can be used with either MACHINE or LOGNAME entries.

NOREAD and **NOWRITE**
Specify exceptions to the READ and WRITE options or defaults.

COMMANDS
Specifies the commands in MACHINE entries that a remote computer can execute on your computer. This affects the security of your system; use it with extreme care.

VALIDATE
Used in conjunction with the COMMANDS option when specifying commands that are potentially dangerous to your computer's security. It provides a certain degree of verification of the caller's identity.

Adding Dial-Out Entries to the Devices File

The **Devices** file (*/usr/lib/uucp/Devices*) contains information for all the devices that can be used to establish a link to a remote computer. Devices are Automatic Call Units, direct links, or network connections. This file works closely with the **Dialers**, **Systems**, and **Dialcodes** files. Before you make changes in any of these files, you should be familiar with them all. A change to an entry in one file may require a change to a related entry in another file.

Each entry in the **Devices** file has the following format:

type ttyline dialerline speed dialer-token

where:

type Can contain one of two keywords (**direct** or **ACU**), the name of a Local Area Network switch, or a system name.

ttyline Contains the device name of the port associated with the **Devices** entry. For example, if the automatic dial modem for a particular entry was attached to the */dev/tty1A* line, the name entered in this field would be *tty1A*.

dialerline This option is useful only for 801 type dialers, which do not contain a modem and must use an additional line. Unless you have an 801 dialer, simply enter a hyphen (-) as a placeholder.

speed is the speed or speed range of the device. Can also contain an indicator for distinguishing different dialer classes.

dialer-token This field contains pairs of dialers and tokens, each representing a dialer and an argument to be passed to it. The *dialer* portion can be the name of an automatic dial modem, or **Direct** for a direct link device.

The Type Field

This field can contain one of two keywords (**Direct** or **ACU**), the name of a Local Area Network switch, or a system name:

Direct This keyword indicates a direct link to another computer or a switch for **cu** connections.

ACU This keyword indicates that the link to a remote computer is made through an Automatic Call Unit. This modem can be connected either directly to your computer or indirectly through a Local Area Network (LAN) switch.

LANswitch can be replaced by the name of a LAN switch. **micom** and **develcon** are supplied with caller scripts in the **Dialers** file.

sysname indicates a direct link to a particular computer. (*sysname* is replaced by the name of the computer.) This means that the line associated with this **Devices** entry is for a particular computer in the **Systems** file.

For example the keyword "gorgon" used in the *Type* field **Devices** is matched against the third field of the **Systems** file entry:

```
Devices:   gorgon ttyla - 1200 hayes1200

Systems:   gorgon Any ACU 1200 14087672676 ogin: nuucp \
             ssword: DontLook
```


The Speed Field

In most cases, this is simply the speed of the device, if the keyword **ACU** or **Direct** is used in the *type* field. However, *speed* can contain a letter and a speed (for example, C1200, D1200) to differentiate between classes of dialers (Centrex or Dimension PBX). This is necessary because many larger offices may have more than one type of telephone network: one network may be dedicated to serving only internal office communications, while another handles the external communications. It is necessary to distinguish which lines are used for internal communications and which are used for external communications. The keyword used in the *speed* field of the **Devices** file is matched against the fourth field of **Systems** file entries, for example:

```
Devices:   ACU ttyla - D1200 hayes1200

Systems:   gorgon Any ACU D1200 3251 ogin: nuucp \
             ssword: DontLook
```

Some devices can be used at any speed, so the keyword **Any** can be used in the *speed* field. If **Any** is used, the line matches any speed requested in a **Systems** file entry. If this field is **Any** and the **Systems** file *speed* field is **Any**, the speed defaults to 1200 bps. If a device can be used at a range of speeds, then the speed field can specify this range (for example, 1200-9600 or D1200-9600). This is preferable to the use of **Any**.

The Dialer-Token Field

NOTE: For best results, dialer programs are preferred over **Dialers** entries. The following entry is an example of an entry using a dialer binary:

```
ACU  ttynn  -  300-2400  /usr/lib/uucp/dialHA24
```

The following binary types are provided in *usr/lib/uucp*:

Binary File	Modem
dialHA12	Hayes Smartmodem 1200 or compatible
dialHA24	Hayes Smartmodem 2400 or compatible
dialVA3450	Racal Vadic 3451 modem
dialTBIT	Telebit Trailblazer Modem

The source is provided for these dialer binaries; you can adapt and compile your own dialers if desired.

Structuring Dialer-Token Entries

The *dialer-token* can be structured four different ways, depending on the device associated with the entry:

- **Simple modem connection.** If an automatic dialing modem is connected directly to a port on your computer, the *dialer-token* field of the associated **Devices** file entry only has one pair. This pair would normally be the name of the modem. This name is used to match the particular **Devices** file entry with an entry in the **Dialers** file. Therefore, the *dialer* field must match the first field of the following **Dialers** file entry:

```
Devices:   ACU ttyla - 1200 ventel

Dialers:   ventel =&-% "" \r\p\r\c $ <K\T%%\r>\c ONLINE!
```

Notice that only the *dialer* portion (**ventel**) is present in the *dialer-token* field of the **Devices** file entry. This means that the *token* to be passed on to the dialer (in this case the phone number) is taken from the *Phone* field of a **Systems** file entry. (\T is implied; see the last item, "Modems used with a local network switch.") Backslash sequences are described later.

- **Direct links.** If a direct-link is established to a particular computer, the *dialer-token* field of the associated entry contains the keyword **direct**. This is true for both types of direct link entries, **direct** and *sysname* (refer to discussion on the *type* field).

- **Local network switches.** If a computer that you wish to communicate with is on the same local network switch as your computer, your computer must first access the switch and the switch can make the connection to the other computer. In this type of entry, there is only one pair. The *dialer* portion is used to match a **Dialers** file entry following:

```
Devices:   develcon tty13 - 1200 develcon \D

Dialers:   develcon "" "" \pr\ps\c est:\007 \E\D\e \007
```

As shown, the *token* portion is **\D**, which indicates that it is retrieved from the **Systems** file without translation. The **Systems** file entry for this particular computer will contain the token in the *phone* field; this is normally reserved for the phone number of the computer (refer to **Systems** file, *phone* field). The **\D** ensures that the contents of the *phone* field is not interpreted as a valid entry in the **Dialcodes** file.

- **Modems used with a local network switch.** If an automatic dialing modem is connected to a switch, your computer must first access the switch and the switch will make the connection to the automatic dialing modem. This type of entry requires two *dialer-token-pairs*. The following *dialer* portion of each pair (fifth and seventh fields of entry) are used to match entries in the **Dialers** file:

```
Devices:   ACU tty14 - 1200 develcon vent ventel

Dialers:   develcon "" "" \pr\ps\c est:\007 \E\D\e \007
           ventel =&-% "" \r\p\r\c $ <K\T%%\r>\c ONLINE!
```

In the first pair, **develcon** is the switch and **vent** is the token that is passed to the **develcon** switch to tell it which device to connect to your computer. This token would be unique for each LAN switch because each switch can be set up differently. Once the ventel modem is connected, the second pair is accessed, where ventel is the dialer and the token is retrieved from the **Systems** file.

The following are two escape characters that can appear in the *dialer-token* field:

\T Indicates that the *Phone* field should be translated at this stage, using the **Dialcodes** file. This escape character is normally placed in the **Dialers** file for each caller script associated with an automatic dial modem (penril, ventel, and so on). The translation will not take place until the caller script is accessed.

\D Indicates that the *Phone* field should not be translated using the **Dialcodes** file. If no escape character is specified at the end of a **Devices** entry, **\D** is assumed by default when a **Dialers** script is to be used (which can itself contain a **\T** to translate the number). **\T** is assumed if a built-in or dialer binary is to be used (because there is then no later opportunity to translate the number).

Using the Same Port for Dialing In and Out

It is possible to dial in and out on the same line without enabling/disabling the line or running a special version of **getty**. All that is necessary is to first create an entry for a line in the **Devices** file (dial-out) and then an entry in */etc/inittab* (dial-in) for the same line. When access to a dial-out line is requested on a shared port, **getty** runs a special program, **uuchat**, that automatically reinitializes the port when the call is complete. **uuchat** uses special dialer scripts found in the **Dialers** file that begin with an ampersand. This means there are actually two entries for some dialers. For example, the dialer for the Hayes Smartmodem 2400 (or compatible) consists of two entries: **hayes2400** and **&hayes2400**, the latter of which is used when reinitializing a shared port to dial-in. In the case of the dialer binaries in */usr/lib/uucp*, these programs are automatically invoked with the **-h** (hangup) switch that reinitializes the port to dial-in.

Administering Your UUCP System

This section discusses the various shell scripts that are used to supervise and maintain UUCP. Consult the section on "Administration and Maintenance Commands" for details on all commands available to the system administrator. Included is an extended description of the */usr/spool/uucp* work directory and a special subsection on troubleshooting.

UUCP Maintenance Shell Scripts

There are several aspects of system operation that are governed by shell scripts running as daemons:

- How often the UUCP directory is checked for work (**uudemon.hour**).
- Polling of sites that are passive (do not originate calls) (**uudemon.poll(2)**).
- Sending of status information to the UUCP administrator (**uudemon.admin**).
- Cleaning of the UUCP spool directory (**uudemon.clean**).

These scripts can be customized and are discussed in **uudemon**(ADM).

Generating Log Reports on UUCP Usage: uulog

The **uulog** program displays log information on UUCP usage according to remote machine. All usage of the programs UUCP, **uuto**, and **uux** are logged in special log files, one per machine. See the **uucp**(C) manual page for more information about **uulog**.

The UUCP Spool Directory

The following is a comprehensive discussion of all files and subdirectories of the UUCP spool directory. These files are created in spool directories to lock devices, hold temporary data, or keep information about remote transfers or executions.

TM. (temporary data file)

These data files are created by UUCP processes under the spool directory (i.e., */usr/spool/uucp/system*) when a file is received from another computer. The *system* directory has the same name as the remote computer that is sending the file. The names of the temporary data files have the format:

TM.*pid.ddd*

where *pid* is a process-ID and *ddd* is a sequential three digit number starting at 0.

When the entire file is received, the **TM.***pid.ddd* file is moved to the pathname specified in the **C.***sysnxxxx* file (discussed below) that caused the transmission. If processing is abnormally terminated, the **TM.***pid.ddd* file may remain in the *system* directory. These files should be automatically removed by **uuclean**.

LCK. (lock file)

Lock files are created in the */usr/spool/uucp* directory for each device in use. Lock files prevent duplicate conversations and multiple attempts to use the same calling device. The names of lock files have the format:

LCK..*str*

where *str* is either a device or computer name. These files may remain in the spool directory if the communications link is unexpectedly dropped (usually on computer crashes). The lock files will be ignored (removed) after the parent process is no longer active. The lock file contains the process ID of the process that created the lock. The lock file is always named using the "a" (non-modem control) suffix to avoid possible conflicts if the same line is specified both modem-control and non-modem-control. For example, the lock on */dev/tty1A* is named *LCK..tty1a*.

C. (work file)

Work files are created in a spool directory when work (file transfers or remote command executions) is queued for a remote computer. The names of work files have the format:

C.*sysnxxxx*

where *sys* is the name of the remote computer, *n* is the ASCII character representing the grade (priority) of the work, and *xxxx* is the four-digit job sequence number assigned by UUCP. Work files contain the following information:

- Full pathname of the file to be sent or requested
- Full pathname of the destination or user/filename
- User login name
- List of options
- Name of associated data file in the spool directory. If the **uucp -c** or **uuto -p** option was specified, a dummy name (**D.0**) is used
- Mode bits of the source file
- Remote user's login name to be notified upon completion of the transfer

D. (data file)

Data files are created when it is specified in the command line to copy the source file to the spool directory. The names of data files have the following format:

D.*systmxxxxyyy*

where *systm* is the first five characters in the name of the remote computer, *xxxx* is a four-digit job sequence number assigned by **uucp**. The four-digit job sequence number may be followed by a sub-sequence number, *yyy* that is used when there are several **D.** files created for a work (**C.**) file.

X. (execute file)

Execute files are created in the spool directory prior to remote command executions. The names of execute files have the following format:

X.*sysnxxxx*

where *sys* is the name of the remote computer, *n* is the character representing the grade (priority) of the work, and *xxxx* is a four digit sequence number assigned by UUCP. Execute files contain the following information:

- Requester's login and computer name
- Name of file(s) required for execution
- Input to be used as the standard input to the command string
- Computer and file name to receive standard output from the command execution
- Command string
- Option lines for return status requests

Troubleshooting

The procedures that follow describe how to solve common UUCP problems.

Check for Faulty ACU/Modem

There are two ways you can check if the automatic call units or modems are not working correctly:

- Run **uustat -q**. This command yields counts and reasons for contact failure.
- Run **cu -x9 -l***line*. This permits you to use a specific line and print debugging information during the attempt. Note that this command is only permitted for those who have write access to the **Devices** file, to protect the modem from interference from unqualified users.

Check the Systems File

If you are having trouble contacting a particular machine, ensure that the information in your **Systems** file is current. Some things that could be out of date are:

- Phone number
- Login
- Password

Debug Transmissions

If you are unable to contact a particular machine, you can check out communications to that machine using **uutry** and **uucp**. Do the following:

1. Make contact using this command line:

   /usr/lib/uucp/uutry -r *machine*

   where *machine* is the node name of the problem machine. This command does the following:

   - Starts the transfer daemon (**uucico**) with debugging. You get more debugging information if you are **root**.
   - Directs the debugging output to /tmp/*machine*.
   - Prints the debugging output to your terminal (**tail -f**). Press the **Del** key to end output.

   You can copy the output from /tmp/*machine* if you wish to save it.

2. If **uutry** fails to isolate the problem, attempt to queue a job with the following command:

   uucp -r *file machine!/dir/file*

   where *file* is the file you want to transfer, and *machine* is the machine you want to copy to, and *dir/file* is the destination location on the other machine. (Remember that the ! must be escaped (\!) if you are using **csh**.) The **-r** option will queue a job without starting a transfer.

3. Next, use **uutry** again. If you still cannot solve the problem, you may need to call support personnel. Save the debugging output; it will help diagnose the problem.

Check Basic Information

There are several commands you can use to check for basic communications information:

uuname Use this command to list the machines you are set up to contact.

uulog Use this command to display the contents of the log directories for particular hosts.

uucheck -v Run this command to check for the presence of files and directories needed by **uucp**. This command also checks the **Permissions** file and outputs information on the permissions you have set up.

Keeping Traffic and Congestion under Control

The UUCP filesystem can be choked by traffic if a connection goes down, but unless your site is running a full USENET feed or your system connects with a number of systems, UUCP should prove self-sustaining. If UUCP is used more frequently on your system, this section discusses how to ensure that the system does not become stopped, congested, or affect the general performance of your system.

Crowded Directories and Lack of Space

The **uudemon.clean** script is the best way to prevent the UUCP spool directory from growing too large. To see how much disk storage is currently used by UUCP, use the **du**(C) command:

du /usr/spool/uucp /usr/spool/uucppublic

The current amount of disk space used in each directory is displayed in 512-byte blocks. Divide this number by two for the size in 1K bytes.

The **uudemon.admin and uudemon.clean** scripts send a great deal of mail to the **uucp** account. You should check and clear the mail file periodically.

Running Out of Processes

On systems with a large amount of traffic, you can get error messages indicating that there are too many processes. If you use the **ps**(C) command, you may notice a number of **uucico** or **uuxqt** processes running. You can establish a new limit on the number of these processes by editing the files **Maxuuscheds**(F) and **Maxuuxqts**(F) in */usr/lib/uucp*.

Evaluating Apparent Stoppages

If users complain that UUCP mail is not getting through and the spool directory is filled with old jobs, it is time to check for the source of the stoppage. UUCP provides an extensive set of error messages and log files that should allow you to trace the cause and remedy the situation.

- Use the **uulog**(ADM) command to study traffic on a per-system basis. Error messages in the *.Admin/errors* are called ASSERT errors. These usually involve filesystem problems.
- Find out the status of currently queued jobs using the **uustat -q** command. This command also indicates the number of failed connection attempts.

Error messages are explained in "UUCP Error Messages" in this chapter. Each message is documented with a suggested remedy.

Complete UUCP Examples

This section includes two complete working examples of a UUCP system and the database files.

Example 1: System gomer

The following system (gomer) has:

- 1200 baud modem on tty4B
- direct connection to system (poker) on tty4D for call out only.
- There are three valid uucp logins:

nuucp	The public login for email. No password required.
ubarn	The on site login for system (poker).
upay4	The private login for email and file transfers.

All lines beginning with # are comments and are not required. Most examples are partial listings and may contain other entries. Micnet is not installed. The modem answers at 1200 baud first and is set up for both call in and out.

NOTE: The lines from */etc/passwd* are included here for informational purposes. **Never edit the /etc/passwd file with a text editor**; this could cause serious problems. Always use the **sysadmsh**(ADM) **Accounts→User→Create** or **Accounts→User→Modify** selections to create or alter UUCP login accounts.

/etc/passwd

```
uucp:*:5:5:Uucp admin:/usr/lib/uucp:
nuucp::201:5:public:/usr/spool/uucplogins/nuucp:/usr/lib/uucp/uucico
upay4:*:202:5:private:/usr/spool/uucppublic:/usr/lib/uucp/uucico
ubarn:*:203:5:poker:/usr/spool/uucppublic:/usr/lib/uucp/uucico
```

/etc/group

```
uucp:x:5:uucp,nuucp,ubarn,upay4
```

/etc/systemid

```
gomer
gomer
```

/etc/intttab

```
t4B:2:respawn:/etc/getty tty4B 2
t4b:2:respawn:/etc/getty tty4b m
t4D:2:respawn:/etc/getty tty4D m
t4d:2:respawn:/etc/getty tty4d 2
```

/usr/lib/uucp/Devices

```
# 300-1200 baud hayes 1200 baud modem.
# The Direct tty4b entry is for using cu to call out.
ACU     tty4B  -  300-1200  dialHA12
Direct  tty4b  -  300-1200  dialHA12
poker   tty4d  -  9600      direct
```

/usr/lib/uucp/Permissions

```
# Public uucp login for mail only.
# Can send mail, transfer files to/from uucppublic, and get
# a directory (ls) listing.
LOGNAME=nuucp     MACHINE=OTHER \
       COMMANDS=rmail:ls:uucp \
       READ=/usr/spool/uucppublic:/usr/tmp \
       WRITE=/usr/spool/uucppublic:/usr/tmp \
       SENDFILES=yes   REQUEST=yes
# Private uucp login for mail and file transfer.
# Only dingbat, ogre, grinch, ... can use this login.
LOGNAME=upay4     VALIDATE=dingbat:ogre:grinch:gomer:blitzen \
       COMMANDS=rmail:ls:uucp:who:uux \
       READ=/  WRITE=/ \
       NOREAD=/etc \
       SENDFILES=yes   REQUEST=yes
# Local trusted connection to gomer
# Only gomer can use this login.
LOGNAME=ubarn     VALIDATE=gomer \
       COMMANDS=ALL \
       READ=/  WRITE=/ \
       SENDFILES=yes   REQUEST=yes
```

/usr/lib/uucp/Systems

```
# local calls
dingbat Any ACU 1200 4444444 ogin:-BREAK-ogin:-BREAK-ogin: \
  uubig word: wetrot
# long distance (evening calls only)
grinch Any1800-0700 ACU 2400 18888888 "" \r ogin:-BREAK-ogin: \
  -BREAK-ogin:nuucp
uunet Any1800-0700 ACU 2400 17031111111 ogin:-BREAK-ogin: \
  -BREAK-ogin:xytpq sword: grm5q
# systems that call in as nuucp (for mail) but NOT call out.
daboss Never
sales Never
guru2 Never
```

Example 2: System dingbat

The following system (dingbat) has:

- 2400 baud modem on tty1A.
- There are two valid uucp logins:

nuucp	The public login for email. No password required.
uubig	The private login for email and file transfers.

All lines beginning with # are comments and are not required. Most examples are partial listings and may contain other entries. Micnet is not installed. The modem answers at 2400 baud first and is setup for both call in and out.

/etc/passwd

```
uucp:*:5:5:Uucp admin:/usr/lib/uucp:
nuucp:*:201:5:public:/usr/spool/uucplogins/nuucp:/usr/lib/uucp/uucico
uubig:*:202:5:private:/usr/spool/uucppublic:/usr/lib/uucp/uucico
```

/etc/group

```
uucp:x:5:uucp,nuucp,uubig
```

/etc/systemid

```
dingbat
dingbat
```

/etc/inittab

```
t1A:2:respawn:/etc/getty tty1A 2
t1a:2:respawn:/etc/getty tty1a m
```

/usr/lib/uucp/Devices

```
# 300-2400 baud hayes 2400 baud modem.
# The Direct entry is for using cu.
ACU  tty1A  -  300-2400  dialHA24
Direct  tty1A  -  300-2400  dialHA24
```

/usr/lib/uucp/Permissions

```
# Public uucp login for mail only.
# Can send mail, transfer files to/from uucppublic, and get
# a directory (ls) listing.
LOGNAME=nuucp      MACHINE=OTHER \
       COMMANDS=rmail:ls:uucp \
       READ=/usr/spool/uucppublic:/usr/tmp \
       WRITE=/usr/spool/uucppublic:/usr/tmp \
       SENDFILES=yes    REQUEST=yes
# Private uucp login for mail and file transfer.
# Only ogre, grinch, ... can use this login.
LOGNAME=uubig      VALIDATE=ogre:grinch:gomer:blitzen \
       COMMANDS=rmail:ls:uucp:who:uux \
       READ=/  WRITE=/ \
       NOREAD=/etc \
       SENDFILES=yes    REQUEST=yes
```

/usr/lib/uucp/Systems

```
# local calls
gomer Any ACU 1200 3333333 ogin:-BREAK-ogin:-BREAK-ogin: \
  upay4 word: dryrot
# long distance (evening calls only)
grinch Any1800-0700 ACU 1200 18888888 "" \r ogin: \
  -BREAK-ogin:-BREAK-ogin: nuucp
# systems that call in as nuucp (for mail) but NOT call out.
daboss Never
damgr Never
guru2 Never
```

Sample Commands

Sending mail to another system and have it send the mail back.

```
mail othersystem!mysystem!mylogin        (Bourne/korn shell)
mail othersystem\!mysystem\!mylogin      (C-shell)
```

Printing Your System's full mail address.

```
echo "`uuname -l`\!`logname`"
```

Displaying the Systems You Can Call.

```
uuname
```

Forcing a call to another system and save the debug output in background.

```
/usr/lib/uucp/uucico -r1 -x7 -Sother 2>/tmp/uulog$$ &
```

UUCP Error Messages

This section lists the error messages associated with UUCP. There are two types of error messages. ASSERT errors are recorded in the */usr/spool/uucp/.Admin/errors* file. STATUS errors are recorded in individual machine files found in the */usr/spool/uucp/.Status* directory.

ASSERT Error Messages

When a process is aborted, ASSERT error messages are recorded in */usr/spool/uucp/.Admin/errors*. These messages include the filename, SCCS ID, line number, and the text listed in these messages. In most cases, these errors are the result of filesystem problems. The "errno" (when present) should be used to investigate the problem. If "errno" is present in a message, it is shown as ( ) in this list.

Error Message	Description/Action
CAN'T OPEN	An open( ) or fopen( ) failed. Check for the presence of the file and permissions.
CAN'T WRITE	A write( ), fwrite( ), fprint( ), etc. failed. Check for the presence of the file and permissions.
CAN'T READ	A read( ), fgets( ), etc. failed. Check for the presence of the file and permissions.
CAN'T CREATE	A create( ) call failed. Check permissions.
CAN'T ALLOCATE	A dynamic allocation failed.
CAN'T LOCK	An attempt to make a LCK (lock) file failed. In some cases, this is a fatal error.
CAN'T STAT	A stat( ) call failed. Check for the presence of the file and permissions.
CAN'T CHMOD	A chmod( ) call failed. Check for the presence of the file and permissions.
CAN'T LINK	A link( ) call failed. Check for the presence of the file and permissions.
CAN'T CHDIR	A chdir( ) call failed. Check for the presence of the file and permissions.
CAN'T UNLINK	A unlink( ) call failed.
WRONG ROLE	This is an internal logic problem.
CAN'T MOVE TO CORRUPTDIR	An attempt to move some bad C. or X. files to the */usr/spool/uucp/.Corrupt* directory failed. The directory is probably missing or has wrong modes or owner.
CAN'T CLOSE	A close( ) or fclose( ) call failed.
FILE EXISTS	The creation of a C. or D. file is attempted, but the file exists. This occurs when there is a problem with the sequence file access. Usually indicates a software error.

Error Message	Description/Action
`No uucp server`	A TCP/IP call is attempted, but there is no server for UUCP.
`BAD UID`	The uid cannot be found in the */etc/passwd* file. The filesystem is in trouble, or the */etc/passwd* file is inconsistent.
`BAD LOGIN_UID`	Same as previous.
`ULIMIT TOO SMALL`	The ulimit for the current user process is too small. File transfers may fail, so transfer is not attempted.
`BAD LINE`	There is a bad line in the **Devices** file; there are not enough arguments on one or more lines.
`FSTAT FAILED IN EWRDATA`	There is something wrong with the Ethernet media.
`SYSLST OVERFLOW`	An internal table in *gename.c* overflowed. A big or strange request was attempted.
`TOO MANY SAVED C FILES`	Same as previous.
`RETURN FROM fixline ioctl`	An ioctl, which should never fail, failed. There is a system driver problem.
`BAD SPEED`	A bad line speed appears in the **Devices/Systems** files (Class field).
`PERMISSIONS file: BAD OPTION`	There is a bad line or option in the **Permissions** file.
`PKCGET READ`	The remote machine probably hung up. No action need be taken.
`PKXSTART`	The remote machine aborted in a non-recoverable way. This can generally be ignored.
`SYSTAT OPEN FAIL`	There is a problem with the modes of */usr/lib/uucp/.Status*, or there is a file with bad modes in the directory.

Error Message	Description/Action
TOO MANY LOCKS	There is an internal problem!
XMV ERROR	There is a problem with some file or directory. It is likely the spool directory, because the modes of the destinations were suppose to be checked before this process was attempted.
CAN'T FORK	An attempt to fork and exec failed. The current job should not be lost, but are attempted later (**uuxqt**). No action need be taken.

UUCP STATUS Error Messages

Status error messages are messages that are stored in the */usr/spool/uucp/.Status* directory. This directory contains a separate file for each remote machine that your system attempts to communicate with. These individual machine files contain status information on the attempted communication, whether it was successful or not. What follows is a list of the most common error messages that can appear in these files.

Error Message	Description/Action
OK	Things are OK.
NO DEVICES AVAILABLE	There is currently no device available for the call. Check to see that there is a valid device in the **Devices** file for the particular system. Check the **Systems** file for the device to be used to call the system.
WRONG TIME TO CALL	A call was placed to the system at a time other than what is specified in the **Systems** file.
TALKING	Self explanatory.
LOGIN FAILED	The login for the given machine failed. It could be a wrong login/password, wrong number, a very slow machine, or failure in getting through the *dialer-token* script.

Error Message	Description/Action
CONVERSATION FAILED	The conversation failed after successful startup. This usually means that one side went down, the program aborted, or the line (link) was dropped.
DIAL FAILED	The remote machine never answered. It could be a bad dialer or the wrong phone number.
BAD LOGIN/MACHINE COMBINATION	The machine called us with a login/machine name that does not agree with the **Permissions** file. This could be an attempt to masquerade!
DEVICE LOCKED	The calling device to be used is currently locked and in use by another process.
ASSERT ERROR	An ASSERT error occurred. Check the */usr/spool/uucp/.Admin/errors* file for the error message and refer to the section "ASSERT Error Messages."
SYSTEM NOT IN Systems	The system is not in the **Systems** file.
CAN'T ACCESS DEVICE	The device tried does not exist or the modes are wrong. Check the appropriate entries in the **Systems** and **Devices** files.
DEVICE FAILED	The open of the device failed.
WRONG MACHINE NAME	The called machine is reporting a different name than expected.
CALLBACK REQUIRED	The called machine requires that it calls your system.
REMOTE HAS A LCK FILE FOR ME	The remote site has a LCK file for your system. They could be trying to call your machine. If they have an older version of UUCP, the process that was talking to your machine may have failed leaving the LCK file. If they have the new version of UUCP, and they are not communicating with your system then the process that has a LCK file is hung.

Error Message	Description/Action
`REMOTE DOES NOT KNOW ME`	The remote machine does not have the node name of your system in its **Systems** file.
`REMOTE REJECT AFTER LOGIN`	The login used by your system to log in does not agree with what the remote machine was expecting.
`REMOTE REJECT, UNKNOWN MESSAGE`	The remote machine rejected the communication with your system for an unknown reason. The remote machine may not be running a standard version of UUCP.
`STARTUP FAILED`	Login succeeded, but initial handshake failed. Check communication parameters: data word size, parity, stop bits, etc.
`CALLER SCRIPT FAILED`	This is usually the same as `DIAL FAILED`. However, if it occurs often, suspect the caller script in the **Dialers** file. Use **uutry** to check.

Administering ODT-DOS

ODT-DOS is based on technology developed for Merge 386 by Locus Computing Corporation.

12/21/89-1.0.0D
Processed: Wed Dec 20 10:52:35 PST 1989

Contents

Chapter 1
Introduction

This guide explains how to administer ODT-DOS. Administering ODT-DOS is no different in most respects from administering separate, conventional DOS and UNIX systems. The administrator's responsibilities include installing and maintaining system hardware and software, regularly backing up system data, assisting users, and informing them of changes to the system.

This guide supplements the system administration instructions in your DOS and UNIX documentation. You should be familiar with that documentation because this guide is not a comprehensive description of the system administrator's responsibilities. In general, you administer ODT-DOS by using UNIX procedures to accomplish UNIX tasks and DOS procedures to accomplish DOS tasks.

Who Should Use This Guide

This guide is for the Open Desktop system administrator, the person responsible for maintaining the day-to-day operation of the system. This guide covers only the administrative procedures that are necessary to manage the combined DOS and UNIX environment of Open Desktop. It supplements the documentation on your computer hardware, DOS, and the UNIX System.

Organization of This Guide

This guide has two additional chapters:

Chapter 2, Administering ODT-DOS, tells you how to manage user accounts, set up and administer computer hardware used by DOS, and configure your computer's resources to meet the combined needs of DOS and UNIX System users.

Chapter 3, Installing DOS Applications, provides hints for installing DOS applications for personal or public use, installing copy-protected applications, setting up DOS applications for use from the UNIX shell, and removing DOS applications.

ODT-DOS Guides

Other guides that describe ODT-DOS operation and administration include:

- *Using ODT-DOS* in the *Open Desktop™ User's Guide.*
- The optional Open Desktop documentation.

Installing ODT-DOS

This guide assumes that you have installed ODT-DOS according to the instructions in the *Open Desktop Installation Guide.*

Release Notes

Be sure to read the *Open Desktop Release Notes* for up-to-date information on supported hardware and software, as well as information on product changes since this guide was printed.

Chapter 2
Administering ODT-DOS

This chapter covers the topics that are essential for using your computer as a combined DOS and UNIX machine. The topics include:

- using the **dosadmin** program,
- adding and deleting user accounts,
- administering DOS applications
- administering the system console,
- administering COM ports,
- administering DOS printers,
- backing up the ODT-DOS filesystem,
- administering disk and diskette drives,
- administering the physical DOS partition,
- administering virtual DOS partitions and floppy disks,
- installing plug-in cards,
- making new DOS images, and
- system files affected by system administration.

To administer ODT-DOS effectively, you should be familiar with the contents of *Using ODT-DOS* in the *Open Desktop User's Guide*, in addition to this chapter.

Most of the descriptions in this chapter assume that you are logged in as root or you are the super user. The UNIX # prompt is therefore shown in most examples. Examples that apply to any user are shown with appropriate DOS or UNIX ($) prompts.

Using the dosadmin Program

The **dosadmin** menu system provides an easy way to change the values of three important DOS characteristics: memory, DOS startup files, and DOS device files. Throughout this guide you will use the **dosadmin** menu to perform various system administration tasks.

Adding and Deleting User Accounts

ODT-DOS requires no special procedures for adding or deleting user accounts. Any user with a valid UNIX account can log into ODT-DOS. You are not required to be logged in as root or have any special permissions, for example, to run DOS or use DOS and UNIX commands.

In general, follow the instructions in the *Administering ODT-OS* in the *Administrator's Guide* for adding, deleting, and administering user accounts.

Although no special configuration files or setup procedures are required to use DOS, Open Desktop users may want to customize the way DOS runs. Users who follow the instructions in *Using ODT-DOS* can alter the behavior of DOS in many ways without help from the system administrator. If users at your site are unfamiliar with the DOS or UNIX systems, however, they may require your assistance. *Using ODT-DOS* contains hints for system administrators who want to modify ODT-DOS defaults or help users configure their own individual environments.

Administering DOS Applications

Occasionally you may run into problems with DOS applications running exactly as they were intended under ODT-DOS. This section covers methods that you can use to fix these problems.

Keyboard Buffer and DOS Applications

Some DOS applications that buffer keystrokes may not work as expected in the ODT-DOS environment. These applications include applications, such as SuperKey, that create keyboard macros by mapping multiple keystroke to a single key.

You can correct this problem by putting the line:

```
device=\usr\dbin\ansi.sys
```

in a CONFIG.SYS file that is interpreted when you run DOS. This approach has a side effect, however. To improve system response, ODT-DOS, by default, puts DOS applications that poll the keyboard to sleep while they wait for keyboard input. When you include the *device=\usr\dbin\ansi.sys* line in your CONFIG.SYS file, ODT-DOS no longer puts applications that poll the keyboard to sleep. Your computer is therefore likely to be more heavily loaded, especially if you run multiple DOS applications at once.

Applications that Poll the Keyboard

DOS applications that poll the keyboard can consume system resources even when they are idle by entering a polling loop. If not compensated for, these applications reduce system performance because when they poll the keyboard, less CPU time is available to other concurrently running processes. By default, ODT-DOS corrects this problem by putting many applications that enter a polling loop to sleep until there is keyboard input.

This method works poorly unless you disable the **pollsleep** feature. To disable **pollsleep**, type at the DOS prompt:

merge set pollsleep off

To restore the default condition, type:

merge set pollsleep omerge set pollsleep on

Network Applications and DOS Drives

Some DOS applications, especially applications designed to work in a network environment, may not recognize ODT-DOS drives that access the shared DOS/UNIX file system (that is, drives C:, D:, or J:) as valid DOS drives.

By default, ODT-DOS treats these drives and files that reside on them as "remote" when an application queries the operating system for information about local and remote drives or files. If a DOS application expects these ODT-DOS drives or files to be "local," the application may fail. If you encounter such a failure, try using the **merge set drive local** or **merge set handle local** command.

The **merge set drive local** command causes ODT-DOS to treat drives that access the shared DOS/UNIX file system as "local." To use this command, start a DOS environment and type the following at your DOS prompt:

```
merge set drive local
```

The command is effective for the duration of the DOS environment. If you want to switch back to the default treatment of drives without exiting the DOS environment, type:

```
merge set drive remote
```

The **merge set handle local** and **merge set handle remote** commands work like the **merge set drive local** and **merge set drive remote** commands, but they cause DOS to interpret files rather than drives as either local or remote. Use the **merge set drive** and **merge set handle** commands individually or in combination according to the requirements of your DOS applications.

Administering the System Console

ODT-DOS works with any system console that consists of a monochrome or color monitor with a PC-compatible keyboard, connected to a monochrome, Hercules, CGA, EGA, or VGA display adapter. *Using ODT-DOS* describes how to use the console from the point of view of the ODT-DOS user. This section tells you how to connect and configure the console. The topics covered here are:

- Setting Up the Console
- Changing Console Display Adapter Cards
- Using Extended Video Modes
- Using the UNIX MultiScreen™ Facility with DOS
- Redefining the ODT-DOS Switch-Screen Key Sequence

Setting Up the Console

No special procedures are required to set up the system console for use with ODT-DOS. The only requirement for using ODT-DOS is that you must use one of the display adapters that ODT-DOS recognizes. These include monochrome, Hercules, CGA, EGA, and VGA adapters. Refer to the *Open Desktop Release Notes* for a specific list of supported display adapters.

ODT-DOS and Display Modes

When you use ODT-DOS, you can run DOS applications on ASCII or PC scancode terminals or on a console that uses a monochrome, Hercules, CGA, EGA, or VGA display adapter and a compatible display. In addition, by using different DOS *images*, ODT-DOS can simulate the characteristics of display adapters less powerful than your actual physical display adapter. (Refer to "Making New DOS Images" later in this chapter for more information on DOS images.)

This flexibility can lead to unexpected problems if you are not careful how you install, configure, and run graphics applications. For example, an application may fail to run if you have configured it to use a different type of display than you really use. If your system uses more than one type of display, you may find that some applications run correctly on some displays but not on others.

Some applications automatically detect the type of display adapter being used and adjust themselves to display graphics data correctly. Other applications do not work properly unless you identify your display adapter when you install them. When you install these applications, be sure you follow the application manufacturer's instructions and configure the applications to work with the display adapter you intend to use.

If you run DOS processes from an ASCII terminal, identify your display as a monochrome display adapter (MDA). If you use a graphics display adapter, you should normally configure your DOS applications to use the type of adapter in your computer, provided the application is compatible with that type of adapter. For example, if you use a VGA card, configure your DOS applications to use a VGA card.

In rare cases, you may run into difficulty because you are trying to use an incompatible combination of physical display adapter, DOS image, and application. By default, ODT-DOS uses DOS images that correspond to your physical display — VGA images for VGA displays, CGA images for CGA displays, and so on. Most DOS applications that are configured for less powerful display adapters run correctly, automatically, when you run them on more powerful adapters. For example, an application configured for CGA displays

may run correctly on a VGA display. However, some applications fail when run on a more powerful display than they are configured for. If you run into this restriction, you can work around it in either of two ways:

- Reconfigure the application so it expects to be run on the display adapter you are using. For example, if your application expects to be run on a CGA card and you use a VGA card, reconfigure the application to expect a VGA card.
- Use a DOS image that corresponds to the display adapter the application expects to use.

For example, if you have a VGA card but your application expects to use a CGA card, you can start a DOS environment with the command:

dos +acga

Then start your application. If you want to start the application directly from the UNIX shell, you can also use a command such as:

dos +acga *appl*

where *appl* is the command that starts your application. These commands cause ODT-DOS to use a CGA image instead of the default image, and your application views your computer hardware as though you actually have a CGA card.

Note that this technique works only if your physical display is capable of using the DOS image you request. The following table shows usable combinations of physical display adapters and DOS images. Fields with a dash (-) indicate unusable combinations.

	Capability By Physical Display Type				
Option	**Monochrome**	**Hercules**	**CGA**	**EGA**	**VGA**
None	mono	herc	CGA	EGA	VGA
+amono	mono	mono	mono	mono	mono
+aherc	-	herc	-	-	-
+acga	-	-	CGA	CGA	CGA
+aega	-	-	-	EGA	-
+arega	-	-	-	EGA	-
+avga	-	-	-	-	VGA

Refer to *Using ODT-DOS* for more information on these options and the characteristics of graphics displays.

Changing Console Display Adapter Cards

If you change your console display adapter card after you install ODT-DOS, you may need to make a new DOS image. See "Making New DOS Images," later in this chapter, for further information.

Using Extended Video Modes

Whether or not you can use extended video modes on EGA and VGA cards depends on how your DOS images were created. If you have an EGA or VGA display and want to run high-resolution graphics applications, you need to create an EGA or VGA DOS image. You can run DOS applications that require monochrome or CGA displays using your EGA or VGA card even if you do not make an EGA/VGA image.

To make an EGA/VGA image, your computer must have a 100 percent IBM-compatible EGA or VGA card. If your card is not fully IBM-compatible, the procedure for make an EGA or VGA image may fail. When the procedure fails, symptoms range from an error message stating that an image cannot be created to locking up the system. Refer to the lsit of supported display adapters in the *Open Desktop Release Notes* for information on tested and certified EGA and VGA cards.

For information on how to create a new DOS image, refer to "Making New DOS Images," later in this chapter.

Using the UNIX MultiScreen Facility with DOS

ODT-DOS is compatible with the standard UNIX MultiScreen facility. If your console can display multiple, independent screens, you can run DOS commands and applications or a DOS environment in any screen.

When you run multiple DOS processes in separate screens, the default key sequence for switching between screens is **Alt-F***n*, where F*n* is a function key (such as F1 or F2). **Alt-F2** switches to screen 2, **Alt-F3** switches to screen 3, and so on. The UNIX system also uses the **Alt-F***n* convention for switching between screens, so you can switch between any active DOS and UNIX screens using **Alt-F***n*.

Redefining the ODT-DOS Switch-Screen Key Sequence

You can redefine the switch-screen key sequence that applies to DOS screens if **Alt-F***n* does not suit your needs. For example, you may have a DOS application that does not work properly with the default switch-screen sequence because it assigns **Alt-F***n* a special meaning. You can redefine the switch-screen key sequence to use a function key together with any combination of the **Ctrl**, **Alt**, and **Shift** keys, or to use only a function key. To redefine the switch-screen key sequence, use the **switchkey** command with the syntax:

switchkey [**-cas**]

In this syntax, **c** stands for the **Ctrl** key, **a** stands for the **Alt** key, and **s** stands for the **Shift** key. To switch screens any time after you use the **switchkey** command, use the keys you specified with **switchkey** in addition to a function key. For example, to specify that you want to require **Ctrl** and **Shift** along with a function key, type:

$ **switchkey -cs**

Thereafter, to switch screens, press **Ctrl-Shift-F***n*.

To specify that you want to use only function keys, without **Ctrl**, **Alt**, or **Shift**, use **switchkey** with only a hyphen as an argument:

$ **switchkey -**

Thereafter you can use F1, F2, F3, and so on, without additional keys, to switch from your current DOS screen to a different one. The **switchkey** command with no arguments displays the current switch-screen key sequence.

The switch-screen key sequence you define with **switchkey** applies only when you are viewing a DOS screen. When you are viewing a UNIX screen, the existing UNIX switch-screen sequence is effective.

ODT-DOS and the X Window System both use the same default switch-screen key sequence and are affected the same way when you use **switchkey**.

Any switch-screen key sequence that you define with **switchkey** applies to any DOS or X Window sessions run on the console where you ran **switchkey**. DOS and X Window sessions that are currently running when you redefine the switch-screen key sequence are also affected. Your specified key sequence remains effective until you issue another **switchkey** command or reboot your computer. To make sure your preferred key sequence remains effective even after a reboot, include the **switchkey** command in */etc/profile* or your home directory *.profile*.

Administering COM Ports

On Open Desktop, both the UNIX environment and DOS can use COM (serial) ports, but only one process (either UNIX or DOS) can access a particular COM port at one time. If you plan to use a COM port for DOS work, follow these procedures:

1. Make sure no **getty** is running on the physical device (for example, */dev/tty1a* or */dev/tty2a*) that you want to make available to DOS as a COM port. To do this, log in as root or become the super user and run the **disable** command. For example, to disable the **getty** process on */dev/tty2a*, type:

   # disable /dev/tty2a

2. Use the **chmod** command to set permissions of the device to be readable and writable. If you want to use */dev/tty2a*, you would type:

   # chmod 666 /dev/tty2a

The COM port is now ready for use with DOS. To return the COM port for use with UNIX, type:

 # enable /dev/tty2a

For more information on using COM ports with DOS, refer to *Using ODT-DOS*

Administering DOS Printers

Using ODT-DOS describes, from the user's point of view, how to print using DOS commands or applications. This section explains how to set up and administer DOS printers. It covers the following topics:

- configuring the default DOS printer,
- changing the default DOS printer,
- adding DOS printers, and
- administering printers directly attached to DOS.

Configuring the Default DOS Printer

By default, ODT-DOS sends all DOS printing via the UNIX spooler to a printer named **doslp**. The ODT-DOS installation routine automatically attempts to configure your default UNIX printer, if one exists, so it can also be used as the default DOS printer, **doslp**. To determine whether **doslp** has been configured, type the following **lpstat** command:

```
# lpstat -p doslp
```

This command tells you whether or not **doslp** is properly configured. If it is not, you need to configure **doslp** yourself.

To configure **doslp**, follow the instructions for configuring printers in the *Administering ODT-OS*. You need to know the UNIX device name of the printer you want to use as **doslp** (for example */dev/lp0*). Use standard UNIX procedures to perform the following operations:

1. Assign the printer name **doslp** to your preferred printing device.

2. Choose the appropriate printer interface program as described below. (The interface program is also known as a printer model.)

Selecting a Printer Interface Program

ODT-DOS supplies a printer interface program named **dosmodel**. It is the same as the **standard** model, except that it does not print banner pages at the beginning of each print job, and it does not output a form feed at the end of each print job.

If ODT-DOS automatically configured **doslp** during the installation procedure, it used **dosmodel**. Any printer that recognizes the **standard** model can also use **dosmodel** for printing DOS output or any other output that does not require banner pages or form feeds.

You can choose the **standard** model for **doslp** if you prefer to have banner pages and automatic form feeds.

If your printer does not recognize the **standard** model, you should not use either the **standard** model or **dosmodel**. Instead, choose a model that is appropriate for your printer.

Changing the Default DOS Printer

The default DOS printer is always named **doslp**. To change the physical printer that **doslp** refers to, follow the standard UNIX procedures for configuring printers. Assign the name **doslp** to your preferred printing device and choose an appropriate interface program.

Adding DOS Printers

You can use printers other than **doslp** for DOS printing. You can select any convenient printer name and any appropriate interface program when you configure a printer.

To use any available UNIX printer for DOS printing, use the ODT-DOS **printer** command to correlate a print stream with a particular UNIX printer and print command. The syntax for selecting a print stream, a printer, and a UNIX print command is:

printer [*print_stream*] **unix** "*print_command*"

where *print_stream* is LPT1, LPT2, or LPT3 and *print_command* is a UNIX command that processes the specified print stream. If you do not specify a print stream, **printer** assumes LPT1 by default. Use the **printer** command in the DOS environment. Assume, for example, that you want to direct DOS printer output sent to LPT2 to a UNIX printer named "laser." Follow these steps:

1. Start a DOS environment if you haven't already started one.

2. Use the following **printer** command to direct print stream LPT2 to to the printer named "laser":

   C> **printer lpt2 unix "lp -dlaser"**

   In this command, the **-d** ("destination") option to the **lp** command identifies the printer named "laser."

3. To send DOS printer output to the printer, name the DOS print stream in your DOS print command.
   For example:

   C> **copy letter.txt lpt2**

You can direct printer output from DOS applications to any UNIX printer using the same procedures.

If you want a **printer** command to be effective every time you use the DOS environment, you can include it in an AUTOEXEC.BAT file.

Note that the *print_command* that you specify when you use the **printer** command is typically lp -d*printer*, where *printer* is the name of a UNIX printer. However, *print_command* can be *any* UNIX command that you choose to use to process a print stream.

Administering Printers Directly Attached to DOS

Under some circumstances, Open Desktop users may not want to use the UNIX spooling system when they use DOS printing. In these cases, users can attach a printer directly to a DOS process. The printer is then under the direct control of DOS, and not available for UNIX printing. Appendix C in *Using ODT-DOS* explains the procedures that users must follow to attach a printer directly to a DOS process.

If a user wants to attach a printer directly that is currently configured for UNIX printing, you need to disable UNIX printing on that printer. Follow these procedures:

1. Log in as root or become the super user.

2. Disable UNIX printing by using the **disable** command. For example, if you want to disable UNIX printing on the printer named **doslp**, issue the command:

   # disable doslp

   The printer is now available for direct attachment to any user's DOS process. Refer to Appendix C in *Using ODT-DOS* for instructions on directly attaching a printer to DOS.

   While UNIX printing is disabled, users can continue to spool print jobs to that printer. However, the jobs are not printed until the printer is re-enabled for UNIX use.

3. When the user finishes the printing that requires directly attaching the printer to DOS, you can use the **enable** command to re-enable UNIX printing. For example, to re-enable the printer **doslp**, issue the command:

   # enable doslp

   NOTE: A single physical printer may have more than one printer name associated with it. If the UNIX printer you are disabling has more than one UNIX printer name, then, in step 2, you should issue the **disable** *printer_name* command for each printer name correlated with that printer. Similarly, in step 3 of this procedure, you may need to issue multiple **enable** commands.

Backing Up the ODT-DOS Filesystem

To guard against permanently losing important data, you should regularly back up all data on your fixed disk. ODT-DOS requires no special procedures for system backups. Simply use UNIX backup procedures for the shared DOS/UNIX filesystem and DOS backup procedures for the physical DOS partition (drive E:).

Consult your DOS and UNIX documentation for specific procedures recommended for your system.

NOTE: If they are stored on a physical or virtual DOS partition, some copy-protected DOS applications cannot be backed up and restored using the BACKUP and RESTORE commands. For more information, consult your DOS user's manual and the instructions for your applications.

Administering Disk and Diskette Drives

Your ODT-DOS system may have one or more diskette drives that can be assigned to either the DOS or the UNIX environment. A diskette drive mounted as a UNIX device is assigned to the UNIX environment and is not accessible as a DOS diskette drive. If you insert a diskette containing a UNIX filesystem into a diskette drive and mount it as a UNIX device, any files on that diskette become part of the shared DOS/UNIX filesystem. You can then use DOS drive C: or J: to access files on the mounted diskette in the same way you access files in the shared DOS/UNIX filesystem on the fixed disk.

To be directly accessible to DOS as DOS drives A: or B:, diskette drives must not be mounted as UNIX devices or in use by UNIX programs (such as **cpio**). If you have more than one diskette drive on your system, you must not have any of them mounted as UNIX devices or otherwise in use by the UNIX system when you attempt to access a diskette drive as DOS drive A: or B:. ODT-DOS prevents a DOS process from using all diskette drives as long as any other process is using any diskette drive.

Sharing Diskette Drives

When diskette drives are not mounted as UNIX devices or otherwise used by the UNIX system, DOS users can access them on a first-come, first-served basis. Whenever a diskette drive has not been used for five seconds or more, it is available to the first DOS process that accesses that drive.

Administering the Physical DOS Partition

The physical DOS partition is a portion of the fixed disk formatted under DOS and reserved exclusively for DOS files. You use the DOS partition under ODT-DOS as DOS drive E:. A physical DOS partition is useful on Open Desktop for the following reasons:

- Some copy-protected DOS applications that cannot be installed in the shared DOS/UNIX filesystem can be successfully installed in the DOS partition.
- You can use files and applications contained in the DOS partition under "raw" DOS, that is, by shutting down the UNIX System and booting DOS.

If you do not already have a physical DOS partition and you choose not to create one, you can still use all ODT-DOS features except drive E:. You can also create and use a virtual DOS partition any time after you install ODT-DOS without removing or reinstalling any files that are currently on your fixed disk. Virtual DOS partitions offer advantages similar to those of a physical DOS partition. Refer to "Administering Virtual DOS Partitions and Virtual Floppy Disks," later in this chapter, for more information.

This section tells you how to create, format, and administer the physical DOS partition.

Creating and Formatting the Physical DOS Partition

If you want to have a physical DOS partition on your Open Desktop computer, you should create one at the time you install the UNIX system, before you install ODT-DOS. Use the utilities packaged with your computer hardware to create and format a DOS partition. Your DOS partition should have a minimum size of 2.5 megabytes. Some DOS copy-protection schemes will not install on a partition smaller than 2.5 megabytes. If you have already installed ODT-DOS, do not have a physical DOS partition, and wish to create one, you may have to back up your fixed disk, create the DOS partition, and reinstall your UNIX system and all applications. Refer to the documentation packaged with your computer for further information.

Default Protection of the DOS Partition

The UNIX filesystem protection mechanisms apply in only a limited way to the DOS partition. Because it is not a UNIX filesystem, access to individual DOS files in the partition is not governed by UNIX user or group ownership or by UNIX read, write, or execute permissions.

However, the DOS partition itself is accessed as a UNIX file or device and can be protected like any other UNIX file or device.

Following are the considerations that affect access to and administration of the DOS partition:

- Although multiple users can read files on the DOS partition at the same time, only one DOS process can write to the DOS partition at one time. In addition, when any DOS process writes to the DOS partition, no other DOS processes can read files on the partition until the process that is writing exits.

- The DOS partition contains a DOS filesystem and is located on a portion of the fixed disk that is physically distinct from the shared DOS/UNIX filesystem.

- The DOS partition is accessed as a UNIX device (sometimes also called a special file). The device name is */dev/dsk/dos*. This device appears to the UNIX system as a single UNIX file, but it can contain within it any number of DOS files and directories.

- When you install ODT-DOS, the DOS partition (that is, the special file */dev/dsk/dos*) is owned by root and is readable and writable by everyone. You can check the ownership and permissions of the DOS partition by typing:

  ```
  # ls -l /dev/dsk/dos
  ```

  The system then shows root as the owner and displays the following permissions:

  ```
  brw-rw-rw-
  ```

  The initial "b" identifies **/dev/dsk/dos** as a block-type special file (as opposed to character-type). UNIX execute permission for **/dev/dsk/dos** is not necessary.

Although it is owned by root, the DOS partition by default is considered to be a public resource that any user can access. All users have read permission so they can inspect the contents of the partition or execute DOS programs stored there. All users have write permission so they can install DOS applications on the partition. Write permission is granted to all users for an additional reason: even if users do not need to create files within the partition, some DOS applications require write permission when they run and will not work if run from an unwritable filesystem.

Changing the Protection of the DOS Partition

If you need to restrict access to the DOS partition, you can use one or both of the following methods:

- Use the DOS ATTRIB command to make specific files in the DOS partition read-only (that is, remove DOS write permission).
- Change the UNIX permission mode of */dev/dsk/dos* to restrict access to the partition.

Using the DOS ATTRIB command to make a DOS file read-only reduces the risk of that file being accidentally removed or corrupted. The protection offered by DOS ATTRIB is limited, however, because anyone who can access a DOS file can also use ATTRIB to make the file writable.

To make a DOS file read-only, use the ATTRIB command to assign the read-only (**R**) attribute as follows:

E> **attrib +r** *filename*

When the read-only attribute is assigned, the file cannot be deleted or changed. To make the file writable again, type:

E> **attrib -r** *filename*

If you want to restrict access to the DOS partition in a more general way, you can use the UNIX command **chmod** to assign any desired permission mode to */dev/dsk/dos*. For example, to deny access to users outside the group the partition belongs to, you could type:

chmod o-rw /dev/dsk/dos

Administering Virtual DOS Partitions and Virtual Floppy Disks

The physical DOS partition is a section of the fixed disk that is formatted under DOS and reserved for DOS files. ODT-DOS also allows you to create virtual DOS partitions, which are UNIX files formatted as DOS volumes and used to store DOS files. You can use a virtual partition like a physical DOS partition to install copy-protected DOS applications that cannot be installed on the shared DOS/UNIX filesystem. Virtual partitions have the following additional advantages:

- You can create a virtual DOS partition easily at any time after you install ODT-DOS. You can therefore avoid the time-consuming process of reconfiguring your fixed disk to create a physical DOS partition if your system does not already have one.

- Users can create and own virtual partitions. By using personal virtual partitions, they can avoid ownership and permission mode conflicts that may arise when all users share the physical DOS partition.

Virtual DOS partitions differ from physical DOS partitions in the following ways:

- Virtual partitions are actually part of the UNIX filesystem. They are UNIX files that contain DOS filesystems. Because virtual partitions are really UNIX files, their contents can be backed up or restored along with other UNIX files using standard UNIX backup commands.

- You cannot access a virtual partition when you have shut ODT-DOS down and booted standard DOS.

After you create a virtual partition, you can attach it as a DOS drive (such as E:).

ODT-DOS also allows you to create a UNIX file and format it under DOS for use as a virtual DOS floppy drive. You can use virtual floppies much as you would use physical DOS diskette drive A: or B:.

The following sections show how to create and administer virtual DOS partitions IX "DOS" "partition" "virtual" and how to create virtual floppies.

Using dosadmin to Create a Virtual DOS Partition

To create a virtual DOS partition, use the following **dosadmin** procedure:

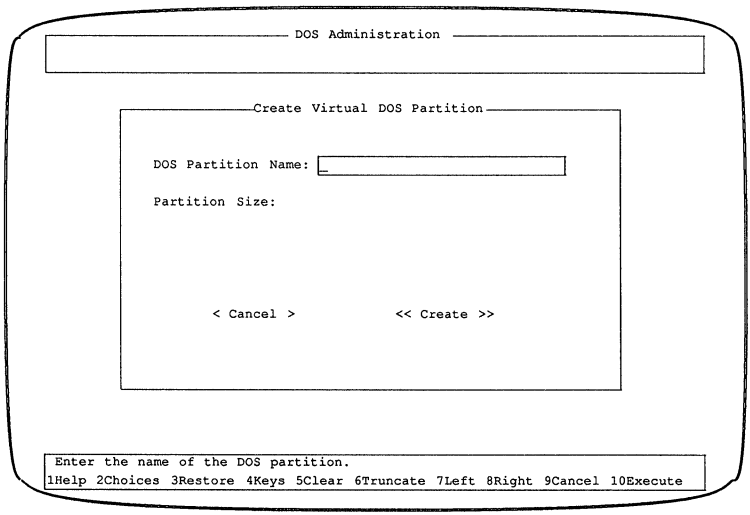
1. Start **dosadmin** by typing:

   $ **dosadmin**

   (You may start **dosadmin** from either the DOS or UNIX command line.) The **dosadmin** ''DOS Administration'' main menu appears.

2. Press the **Left** or **Right Arrow** key (← or →) to move the highlighted field to ''Merge''.

3. Press the **Up** or **Down Arrow** key (↑ or ↓) to move the highlighted field to the ''Create Virtual Partition'' field. Press **Enter** (↵) to select this item and display the following screen:



4. In the "DOS Partition Name" field, type the full path of the file you want to use as your virtual DOS partition. You supply the pathname using either DOS style (including the DOS drive name and using backslashes as path separators) or UNIX style. For example, to create a virtual DOS partition named */usr/joe/vpart*, type:

   /usr/joe/vpart

   Then press **Tab** (⇆) to move the highlight to the "Partition Size" field.

5. Enter the size of the virtual partition, in kilobytes (1024 kilobytes equals 1 megabyte; the recommended minimum size is 2.5 megabytes). For example, to create a 2.5-megabyte virtual DOS partition, type:

   2560

   Then press **Tab** to move to the "< Cancel >" field.

6. If you choose to cancel the operation and not create the virtual partition, press **Enter** while "< Cancel >" is highlighted. Otherwise, check your entries for accuracy. If any information on the screen is incorrect, press **Tab** to move to the field you want to change and retype the entry. (Press **F5** to clear a field completely.)

   When the information is correct, press **Tab** to highlight the "<< Create >>" field and then press **Enter**. ODT-DOS creates the virtual DOS partition you have selected.

Using Virtual Partitions

To use a virtual DOS partition, you must attach it as a drive between E: and Z: when you start DOS. For example, to use the */usr/joe/vpart* DOS partition as drive F:, you would start the DOS environment by typing:

```
$ dos +af:=/usr/joe/vpart
```

For the duration of the DOS environment, */usr/joe/vpart* is accessible as DOS drive F:. You can use the **dosopt** command to automatically attach a DOS partition whenever you run DOS. Refer to the descriptions of the **dosopt** command and the **±a** option in Appendix A of *Using ODT-DOS* for further information.

Note that the DOS drive name for a physical DOS partition, if you have one, is "E:". Only one partition at a time is available as drive E:. If you have a physical DOS partition, it is available by default as drive E: when you start a DOS process. If you want to use a virtual partition as drive E:, you can use the command **dos +ae:=***pathname* whether or not you have a physical DOS partition.

You should not use drive J: to attach a virtual partition since drive J: has a specific function in Open Desktop. Drive J: works exactly like drive C:, but you can have different current directories on the two drives. You can also use drive J: to simplify the use of some older DOS applications that require both the application and data or text files to be in the current directory.

Also, when specifying a drive name, be sure you redefine LASTDRIVE in your CONFIG.SYS file if you use a drive letter late in the alphabet. The default LASTDRIVE is "N".

Administering Virtual DOS Partitions

Most of the information covered under "Administering the Physical DOS Partition" in this chapter also applies to virtual DOS partitions. Virtual DOS partitions differ from physical DOS partitions in the following ways:

- A virtual DOS partition is a single file within the shared DOS/UNIX filesystem. It contains a DOS filesystem within it. It is not useful from the UNIX environment. You can use standard UNIX tools to move it around, back it up, or delete it. Deleting it frees up space on the shared DOS/UNIX filesystem.

- When you specify the size of a virtual partition at the time you create it, the partition does not immediately consume the disk space you specify. As you add files, the partition expands to consume the disk space required by the files up to the maximum amount you specified when you created the partition.

  Once the space is used, however, you cannot free space in the UNIX filesystem by removing files from the partition. Furthermore, if you back up and restore the partition using UNIX utilities, the partition always consumes the full amount of UNIX filesystem space specified when you created it, whether or not you have filled it with DOS files.

Using dosadmin to Create a Virtual DOS Floppy

Virtual floppy disks are UNIX files that have been formatted under DOS and contain DOS volumes in sizes that correspond to standard DOS diskettes. You may use them just as you would physical diskettes to store DOS files. You may even transfer DOS system files to a virtual floppy (using the DOS SYS command) and boot from it.

NOTE: When you boot from a virtual floppy, you normally cannot access the shared DOS/UNIX filesystem.

To create a virtual DOS floppy, use the following procedure:

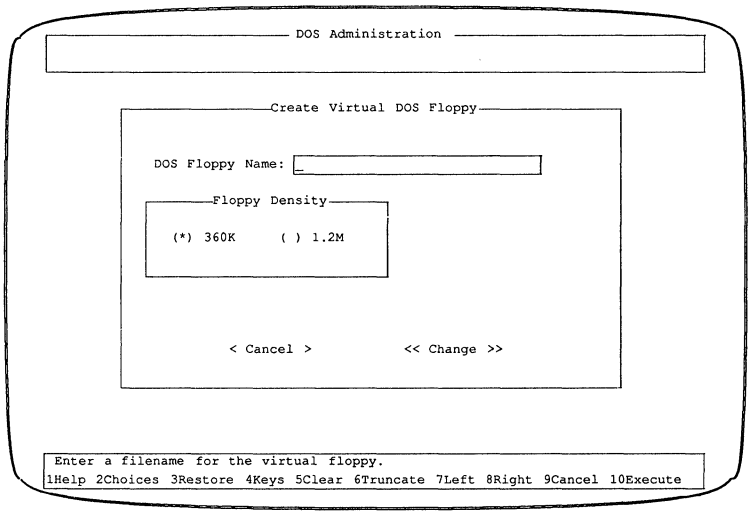
1. Start **dosadmin** by typing:

   $ **dosadmin**

   (You may start **dosadmin** from either the DOS or UNIX command line.) The **dosadmin** "DOS Administration" main menu appears.

2. Press the **Left** or **Right Arrow** key (← or →) to move the highlighted field to "Merge".

3. Press the **Up** or **Down Arrow** key (↑ or ↓) to move the highlighted field to the "Create Virtual Floppy" field. Press **Enter** (↵) to select this item and display the following screen:



4. Enter the full pathname of the file you want to use for your virtual DOS floppy in the ''DOS Floppy Name'' field. You supply the pathname using either DOS style (including the DOS drive name and using backslashes as path separators) or UNIX style. For example, to create a virtual DOS floppy named */usr/joe/vflop*, type:

   /usr/joe/vflop

   Then press **Tab** (⇆) to move to the ''Floppy Density'' field.

5. The field for low-density virtual floppies should be highlighted. This is the default for virtual floppies. To create a low-density floppy, press **Tab** (⇆) to move to the next field.

   If you want to create a high-density floppy, press the **Left** or **Right Arrow** key (← or →) to highlight the high-density value. Press **Enter** or the **Space** to select this option; an asterisk (*) appears between the parentheses. Then press **Tab** (⇆) to move to the next field.

6. If you decide to cancel the operation and not create a virtual floppy, press **Enter** while the ''< Cancel >'' field is highlighted. Otherwise, check your entries for accuracy. If any information on the screen is incorrect, press **Tab** to move to the field you want to change and retype your entry. (Use **F5** to clear a field completely.)

   When the information is correct, press **Tab** to highlight the ''<< Create >>'' field, and then press **Enter**.

ODT-DOS creates and formats the virtual floppy as you have specified. To use the virtual floppy, you must attach it as floppy drive A: or B: when starting DOS. For example, to use the **/usr/joe/vflop** virtual floppy as drive B:, you would start the DOS environment by typing:

$ **dos +ab:=/usr/joe/vflop**

When you attach a virtual floppy like this, you cannot access a physical DOS diskette drive with the same name (drive B: in this example). You can, however, access physical diskette drives with different names.

To boot a virtual floppy, use the **dos -l** (''no load image'') option and use the **+a** option to attach the virtual floppy to your DOS process. The syntax is:

dos -l +a*drive_letter*:=*pathname*

where *drive_letter* is either `a` or `b` and *pathname* is the full path of the UNIX file containing the virtual floppy. For example:

```
$ dos -l +aa:=/usr/joe/vflop
```

Refer to Appendix A in *Using ODT-DOS* for further information on the **±l** option.

Installing Plug-In Cards in Your Computer

ODT-DOS allows DOS to use standard hardware devices in a convenient way. Standard DOS hardware devices include displays, disk drives, COM ports, expanded memory (EMS), a mouse, printer ports, and the game port. A few pointers on administering several of these devices appear earlier in this chapter. Also refer to *Using ODT-DOS* (especially the description of the **±a** option) for information on using standard devices.

This section tells you how to install, configure, and use other DOS devices. DOS can use other devices only by communicating directly with those devices, without the intervention of the UNIX system. This form of device access is called direct attachment.

When you add, remove, or change computer hardware, you should make new DOS images. See ''Making New DOS Images,'' later in this chapter, for more information.

Using Directly Attached DOS Devices

Install your directly attached DOS device according to the manufacturer's instructions.

To use a directly attached DOS device, you must use the **+a** option to attach it to the DOS process when you start DOS. The syntax for direct device attachment is:

+a*device_specification*

where *device_specification* consists of the following fields:

io_port_range (typically in the range of 0-3ff, hex)
interrupt_level (3-7, 9-12, 14, or 15)
memory_mapped_io_range (typically c0000-effff, hex)
dma_channel (0, 1, 2, 3, 5, 6, 7)

The device specification fields must be listed in the order shown, but only those needed must be specified. Fields are separated by dots (.). The first field has a leading dot (between it and **+a**). When a field is not relevant, a single dot is used. An example of a command that starts the DOS environment and directly attaches a device is:

```
# dos +a.2ff.5..
```

Following are more complete descriptions of each of the device specification fields:

- **io_port_range** has this recursive definition:

  X[**-***Y*][**,***io_port_range*]

  where the form *X* specifies an I/O port address *X* (in hex) and the form *X-Y* specifies an I/O port address range, from low *X* (in hex) to high *Y* (in hex), that the device uses. As the syntax shows, single or multiple ports or ranges may be given.

- **interrupt_level** is a single decimal number that denotes which interrupt level is used.

- **memory_mapped_io_range** has this recursive definition:

  X-Y[**,***memory_mapped_io_range*]

  where *X* and *Y* are two hex numbers, separated by a hyphen, denoting the range of the memory-mapped I/O. The first number is the lowest address and the second is the highest address. Each range must start on a 4K boundary and be a multiple of 4K. As the syntax shows, multiple ranges may be specified.

- **dma_channel** is a single decimal number that denotes which DMA channel is used.

Determine the values for **io_port_range, interrupt_level, memory_mapped_io_range**, and **dma_channel** by consulting the hardware technical specifications for the devices being used. If you have difficulty determining the required values for your hardware, contact the hardware manufacturer. Following are examples showing how to use the **+a** option to specify a directly attached device:

- **Directly attached COM1.**[1] The COM1 interrupt is 4, and the ports range from 3f8 to 3ff. The device specification is therefore:

  ```
  .3f8-3ff.4..
  ```

  The last two fields, memory-mapped I/O range and DMA channel, are blank because these fields are not relevant for a COM port. You might use this device specification in commands such as:

  ```
  # dos +a.3f8-3ff.4..
  # dos +a.3f8-3ff.4.. xtalk
  ```

- **A hypothetical hardware device called "widget" that uses memory-mapped I/O in addition to ports and interrupts.** The device has the following characteristics:

  Port ranges: 2f8-2ff and 3f0
  Interrupt channel: 2
  Memory mapped I/O range: c0000-c03ff

  The device specification for this device is:

  ```
  .2f8-2ff,3f0.2.c0000-c03ff.
  ```

  You can attach this device to DOS when you start the DOS environment with the command:

  ```
  # dos +a.2f8-2ff,3f0.2.c0000-c03ff.
  ```

[1] This example is for tutorial purposes only. ODT-DOS is configured by default to attach directly the COM1 and COM2 ports when you use the **+adcom1** or **+adcom2** DOS option.

If you have an application called "colorama" that uses this device, you can invoke the application and attach the device at the same time with the command:

```
# dos +a.2f8-2ff,3f0.2.c0000-c03ff. colorama
```

Incorrect use of device specifications for directly attached devices can crash Open Desktop. Therefore, only the system administrator (logged in as root or super user) is allowed to specify directly attached devices on the command line, as illustrated in the preceding examples. To enable other users to access directly attached devices, you must define tokens for the devices in the *dosdev* file. The next section describes this procedure.

Setting Up /etc/dosdev

The ODT-DOS device specification file */etc/dosdev* serves several purposes. *dosdev* includes specifications for several standard devices, including printers, COM ports, and expanded memory. It also includes comments and example "template" device specifications that you might find useful as you set up *dosdev* specifications for other DOS devices.

Normally, you should not alter existing *dosdev* specifications. You can add your own device specifications to *dosdev*, though. The primary reasons for specifying a directly attached device in *dosdev* are:

- To allow ODT-DOS users to directly attach devices. When you specify a device in *dosdev*, users other than root or super user can directly attach devices.

- To create a short, easily remembered alias (or token) for a device specification. You and other users can use this token with the **dos +a** option instead of fully specifying the device as described in the previous section.

Before specifying a directly attached device in *dosdev*, you should test the device specification using the **+a** option as described in the previous section. Because incorrect use of device specifications for directly attached devices can crash Open Desktop, you should conduct your tests when users will not be inconvenienced if you need to reboot the system.

The syntax for specifying a directly attached device in *dosdev* is:

token **d** *device_specification comments*

Tokens may not contain spaces, commas, or equal signs and may not start with a slash or dot. Tokens also may not resemble a drive letter specification (a single letter followed by a colon). Aside from these restrictions, you can use any alphanumeric characters in tokens. Tokens can be up to 32 characters in length. For example, the *dosdev* line for the directly attached COM1 port discussed in the previous section is:[1]

```
dcom1 d .3f8-3ff.4.. Direct attach COM device.
```

A possible *dosdev* entry you could create for the hypothetical hardware device discussed in the previous section is:

```
widget d .2f8-2ff,3f0.2.c0000-c03ff. Direct attach widget.
```

These lines can be anywhere in *dosdev*. As long as these lines exist in *dosdev*, users can directly attach COM1 or widget with commands such as:

```
$ dos +adcom1
$ dos +awidget
```

Users can also use the **dosopt** command to cause specific applications or the **dos** command to request access to a directly attached device automatically. Use the token defined in *dosdev* when issuing the **dosopt** command. For example:

```
$ dosopt +awidget dosenv.def
```

When you use **dosopt** like this, ODT-DOS looks up specification for the token (''widget'' in this example) in *dosdev* when you run DOS, not when you assign the option with **dosopt**. Therefore, if you change the device specification associated with the token in *dosdev*, ODT-DOS always uses the current specification. Refer to *Using ODT-DOS* for further information on **dosopt**.

Note that directly attached devices cannot be shared. As long as one process is using a directly attached device, other processes are prevented from using the device until the process controlling the device exits.

[1] Recall that this example is for tutorial purposes only. You do not need to add this line to *dosdev*, because it is already there by default.

Mapping Device Names

Open Desktop allows DOS to access a hardware device via different device specifications than the physical device uses. For instance, assume you have a communications program that uses COM1 by default. If you decide you want the application to use the physical COM2 port instead, you can use the mapping feature to make the application treat COM2 as though it were COM1, without reconfiguring the application or physically moving hardware. This feature is useful whenever you want DOS to view a hardware device as though were a different physical device.

To make DOS view a hardware device as though it had different specifications, you map the virtual device specification (the specification that DOS sees) to the physical device specification (the actual physical characteristics of the device). The syntax for mapping device names is:

+a*virtual_device_specification=physical_device_specification*

virtual_device_specification and *physical_device_specification* can be:

- Complete device specifications (including fields for I/O port range, interrupt level, memory-mapped I/O range, and DMA channel).

- Tokens that are defined in *dosdev*.

For example, the tokens **dcom1** and **dcom2** are defined by default in *dosdev*, so you can start the DOS environment and map COM1 to COM2 with the command:

```
$ dos +adcom1=dcom2
```

This command causes DOS to treat the physical COM2 port as though it were COM1.

There is no difference in performance when you map virtual and physical DOS devices like this, except for I/O ports, for which it is more efficient to make the virtual and physical port ranges the same.

Making New DOS Images

The *DOS image* is a file that reflects the configuration of your virtual PC environment. The DOS image is a frozen picture, or snapshot, of DOS after it has been booted and loaded into memory and begun running. This image includes information DOS needs about the system configuration, derived from the hardware installed on the system and the contents of the file */usr/lib/merge/config.sys* at the time the image is made. It allows quick startup of a DOS program from UNIX. The default DOS images (one each for monochrome, CGA, EGA, and VGA display adapters) are contained in the files */usr/lib/merge/mono.img*, */usr/lib/merge/cga.img*, */usr/lib/merge/ega.img*, and */usr/lib/merge/vga.img*.[1] These DOS image files are used by all ODT-DOS users whenever they run any DOS process unless they specifically request a custom image when they start DOS.

There are several reasons to make new DOS images. Typically, when new images are required, the system administrator creates a new default DOS image for each type of display adapter used on Open Desktop. The new default images are then available automatically to all DOS users. The following subsections describe when you need to create new DOS images and how to create them. The topics are:

- Hardware Changes that Require New DOS Images
- Changing the CONFIG.SYS STACKS Command
- Using **dosadmin** to Make New DOS Images

Hardware Changes That Require New DOS Images

The most common reason to make new DOS images is that you have changed your computer hardware. New DOS images are required whenever you make hardware changes that change the BIOS data area or interrupt vectors. You may not know whether a particular hardware change has had these effects. Making new DOS images is simple, however, and it never hurts to make new images even when they are not required. We therefore recommend that you make new default DOS images for all types of displays used on your system after you make any hardware change.

[1] Hercules display adapters also use the */usr/lib/merge/mono.img* file.

Standard ROMs and On-Card ROMs

When you create a DOS image for an EGA or a VGA display adapter card, ODT-DOS uses the data in the display adapter ROM. ODT-DOS can use either the actual physical ROM on the card while making the image, or a standard ROM supplied (as a file) with the ODT-DOS software.

The images created during the ODT-DOS installation procedure use standard ROM files, rather than the on-card ROMs, whenever standard ROMs exist. When you make a new image following the initial ODT-DOS installation, you may be able to choose between a standard ROM and an on-card ROM. When both a standard ROM and an on-card ROM exist for a particular card, you must specify whether you want to use the standard ROM or the on-card ROM. Consider the following trade-offs as you make your choice:

- The image-making process is more reliable when you use standard ROMs. Making an image using some on-card ROMs can fail in unpredictable ways. Symptoms may range from an error message stating that an image cannot be made to the system locking up .

- When you make an image for a display adapter using a standard ROM, you are restricted to using the standard IBM video modes for that card. If you have a display adapter that provides nonstandard functionality and you want to take advantage of that functionality, you must make an image using the on-card ROM.

  NOTE: ODT-DOS may not operate correctly when you use nonstandard video modes. Because extended video modes can vary in unpredictable ways (in their treatment of registers in particular), it is impossible to predict how ODT-DOS will work when you use these modes.

To create a new DOS image, follow the procedures in "Using **dosadmin** to Make New DOS Images," later in this guide.

Changing the config.sys STACKS Command

ODT-DOS allows you to use nearly all *config.sys* commands as you would on a conventional DOS computer. Commands in the root directory *config.sys* file affect all users when they start a DOS process. Commands in a user's home directory *config.sys* file affect only that user's DOS processes.

On Open Desktop, however, DOS does not interpret the *config.sys* STACKS command unless it is incorporated into the DOS image you are using. DOS ignores STACKS commands in the root directory and home directory *config.sys* files and any other *config.sys* files you specify with the **+e** option at DOS runtime.

To incorporate a STACKS command into one or more DOS images, you first include the STACKS command you want to use in the file */usr/lib/merge/config.sys*. This *config.sys* file is used only in the DOS image-creation process. It includes configuration information (such as device driver definitions) that are needed whenever users run DOS processes.

WARNING: Do not alter any of the lines that exist in */usr/lib/merge/config.sys* when you install ODT-DOS. You can edit this file to include any additional *config.sys* instructions that you want incorporated into a DOS image, but altering any of the factory default lines is likely to result in unexpected and undesirable ODT-DOS behavior. The only reason you are likely to need to modify this file is to add or change a STACKS command.

After adding to */usr/lib/merge/config.sys*, create one or more new DOS images by following the instructions in the next section. You need to make a new image for each type of display to which you want the new */usr/lib/merge/config.sys* to apply.

Using dosadmin to Make New DOS Images

To make new DOS images, follow these steps:

1. If you are creating a new image because you have made hardware changes, make sure that the hardware that requires the new DOS image (for example, a new VGA card) has been installed on the system.

2. Log in as root or become the super user.

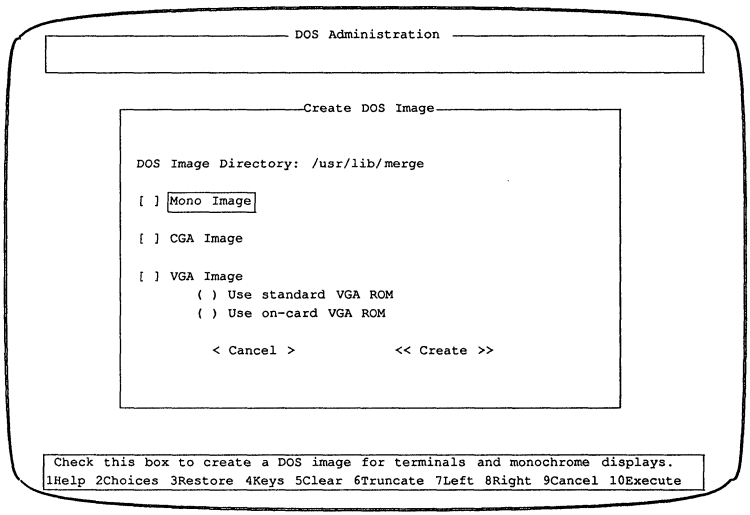
3. Start **dosadmin** by typing:

   # dosadmin

   (You may start **dosadmin** from either the DOS or UNIX command line .) The **dosadmin** "DOS Administration" main menu appears.

4. Press the **Left** or **Right Arrow** key (← or →) to move the highlighted field to "ODT-DOS." A submenu opens under "ODT-DOS."

5. If the "Create DOS Image" field is not highlighted, press the **Up** or **Down Arrow** key (↑ or ↓) to move the highlighted field to "Create DOS Image." Then press **Enter** (↵) to select this item and display the following screen:



The "DOS Image Directory" shows */usr/lib/merge* as the default directory for system images. Below this field are fields for each type of image you can create. These fields vary, depending on your computer hardware. For example, fields for VGA images appear only if you have a VGA card installed.

6. If you want to create DOS images in a directory other than */usr/lib/merge*, enter your preferred directory name in the "DOS Image Directory" field. If this field is not already highlighted when you want to change it, use the **Tab** key (⇆) to move the highlight. The directory must be specified by full pathname and may be entered either in UNIX format, using slashes (/) as path separators, or in DOS format, specifying a DOS drive (C: or D:) and using backslashes (\).

NOTE: To use a DOS image stored in a directory other than */usr/lib/merge*, you must explicitly request the image using the DOS **+l** option when you run DOS. See the description of the **+l** option in *Using ODT-DOS* for further information.

7. Now check the box adjacent to the name of each image you want to create. Use the **Tab** key to move the highlight from one field to another, and press **Enter** or **Space** to check the selection box for each image you want to create. You can select any or all of the displayed image types. Each image you select is created in the directory specified in the "DOS Image Directory" field in a file named to correspond to the image type — *mono.img*, *cga.img*, *ega.img*, or *vga.img*.

8. If your system allows you to create images that use either a standard ROM or an on-card ROM, the **dosadmin** screen offers you a choice between the two types of ROM. The default is to use the standard ROM, so by default this selection has an asterisk next to it. If you want to use the on-card ROM, use the **Down Arrow** key (↓) to move the highlight to the "Use on-card ROM" field. Then press **Enter** or the **Space** to select this field. An asterisk appears between the parentheses.

9. If you decide not to complete this procedure, press **Tab** to move the highlight into the "< Cancel >" field and press **Enter**. Otherwise, to complete the procedure, press **Tab** until "<< Create >>" is highlighted, then press **Enter**.

System Files Affected by System Administration

For your reference, the following is a list of ODT-DOS files affected by common administrative procedures. These system files are used by the general ODT-DOS user community. Most of these files are described earlier in this chapter. Refer to Chapter 3 in this guide for further information on */usr/lib/merge/sdfile*, */etc/dosenv.def*, and */etc/dosapp.def*.

- */autoexec.bat*
- */config.sys*
- */etc/dosenv.def*

- */etc/dosapp.def*
- */etc/dosdev*
- */usr/lib/merge/mono.img*
- */usr/lib/merge/cga.img*
- */usr/lib/merge/config.sys*
- */usr/lib/merge/ega.img*
- */usr/lib/merge/vga.img*
- */usr/lib/merge/sdfile*

Be sure you inform all affected users (via mail or message of the day) whenever you modify any of these files.

Chapter 3
Installing DOS Applications

This chapter tells you how to install and configure DOS applications on the Open Desktop fixed disk and how to remove them. In general, you can install most applications simply by entering the DOS environment as described in *Using ODT-DOS* and then following the application manufacturer's instructions for installing on a fixed disk. ODT-DOS, however, supplies a menu system called **dosadmin** that automatically configures applications so they can be used from both the UNIX shell and the DOS environment. The **dosadmin** utility also provides simple ways of keeping track of installed applications and tailoring them.

A few copy-protected applications cannot be installed on the shared DOS/UNIX filesystem. You can install these applications on the DOS partition as described later in this chapter.

Some applications cannot be installed on the fixed disk at all because they boot their own operating systems from the distributed diskette to run the program. To run these applications, use the ODT-DOS **dosboot** command as described in *Using ODT-DOS*.

This chapter includes the following sections:

- Installing DOS Applications Using **dosadmin**
- Installing Copy-Protected DOS Applications
- Removing DOS Applications

Installing DOS Applications Using dosadmin

If you want to use DOS applications only from the DOS environment, you do not need to read this section. You can start a DOS environment and install your applications according to the manufacture's instructions. The **dosadmin** procedures described here configure DOS applications so they can be used conveniently from the UNIX shell. If you choose to install a DOS application without using **dosadmin**, you can still use **dosadmin** to configure the application for use from the UNIX shell at a later time.

The system administrator can use **dosadmin** to install public applications in a directory accessible to all users. Individual users can use **dosadmin** to install personal DOS applications in their own directories.

You can use **dosadmin** to install most DOS applications that are designed to be installed on a personal computer fixed disk. The exceptions are DOS applications using copy-protection schemes that require installation on an actual DOS filesystem or on a system running DOS as its native operating system. For further information on this subject, see "Installing Copy-Protected Programs," later in this chapter.

Following is an outline of the procedure for installing applications using **dosadmin**:

- After logging into ODT-DOS, type **dosadmin**.
- Select the "Install Applications" menu and answer some pre-installation questions concerning the application.
- Install the application on the fixed disk according to the application manufacturer's instructions.
- Respond to post-installation **dosadmin** prompts concerning system configuration required by the application.

Following are step-by-step instructions for using **dosadmin** to install DOS applications on the Open Desktop fixed disk. Before installing any application, you should read through these instructions to make sure you have the necessary information at hand. You should also be familiar with the application manufacturer's installation instructions.

You follow the same general procedures whether you are the system administrator installing an application in a public directory or a user installing a personal application in your home directory. The next section, "Installing Public DOS Applications," tells the system administrator how to install applications for use by all system users. The following section, "Installing Personal DOS Applications," tells you how to install a personal application in your own directory.

The screen displays and procedures described in these sections use Lotus 1-2-3 as a typical example. Your responses to **dosadmin** prompts and the precise appearance of your display will differ, depending on your application and how you want to configure it.

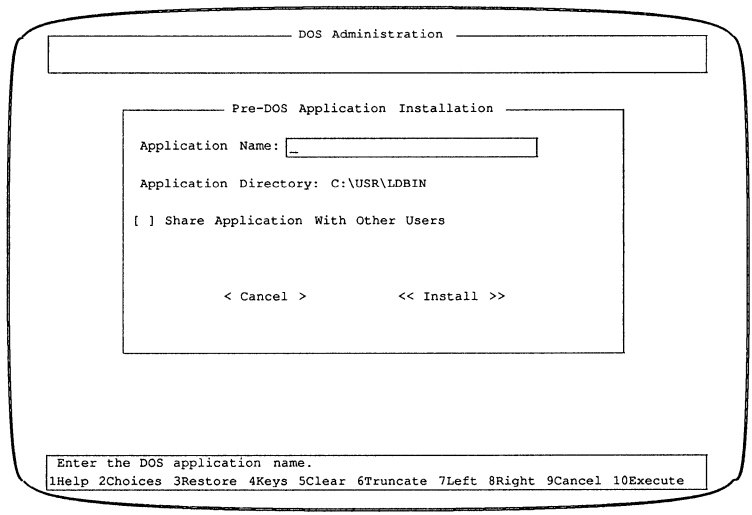
Installing Public DOS Applications

The Open Desktop system administrator can use **dosadmin** to install DOS applications intended for shared use by all system users. To install a public DOS application, follow these procedures:

1. Log in as root.
2. Type:[1]

   # **dosadmin**

   The **dosadmin** main menu is displayed.
3. The "Applications" submenu should be highlighted. If it is not, press the **Left** or **Right Arrow** key (← or →) until it is.
4. Press the **Down Arrow** key (↓) to move the highlight to "Install" and press **Enter** (↵). The "Install Applications" menu appears, and your display looks like this:



[1] You can run **dosadmin** from either the DOS environment or the UNIX shell. If you start **dosadmin** while using the DOS environment, your prompt before starting **dosadmin** and after exiting **dosadmin** is a DOS prompt, such as **C>**. When you start **dosadmin** from the UNIX shell, your prompt before and after using **dosadmin** is **#**. The actual **dosadmin** procedures and displays are identical, whether you start **dosadmin** from the UNIX shell or in the DOS environment.

5. Type in the name of the application you are installing. The name you type here is entered in the application database, allowing you to access the application easily, by name, with the List Applications, Remove Applications, and Tailor Applications menus. The name you type is associated in the database with one executable file. With applications (such as Lotus 1-2-3) that include more than one executable file (LOTUS.COM and 123.COM), it is useful to include the name of the command you normally use with that application. For example, if you normally use Lotus by typing **lotus**, you might enter **Lotus 1-2-3 (lotus)** as the name of the application. If you normally use Lotus by typing **123,** you could enter **Lotus 1-2-3 (123)** as the name of the application.

   If you make a mistake while typing, use the **Bksp** key (←) to back up and correct it, or use the **F5** key to clear the field and retype your entry.

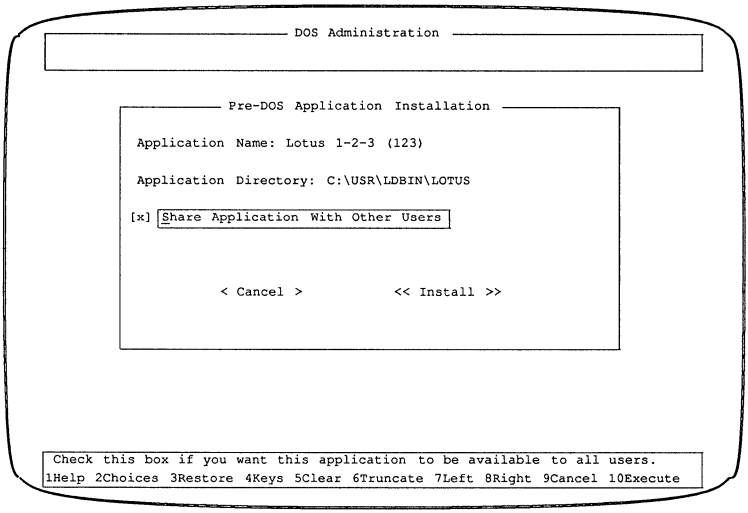
6. Press the **Tab** key (⇆) to move to the "Application Directory" field. "C:\USR\LDBIN" is already filled in. We recommend you install public DOS applications in this default directory.

7. If you want to install the application in \USR\LDBIN (the default), skip to step 8.

   If you want to change the default drive or directory for installing the application, enter your choice here. You may, for example, want to install the application in a subdirectory, such as \USR\LDBIN\LOTUS. To specify a subdirectory like this, use the right arrow key (→) to move the cursor to the right of the displayed "C:\USR\LDBIN", and type in the name of the subdirectory. If the directory you specify does not already exist, **dosadmin** creates it during the installation procedure.

8. Press the **Tab** key (⇆) to highlight the "Share Applications With Other Users" field.

9. Because you are installing a public application, press the **Space** or **Enter** key to check the "Share Application with Other Users" box. The display looks like this:

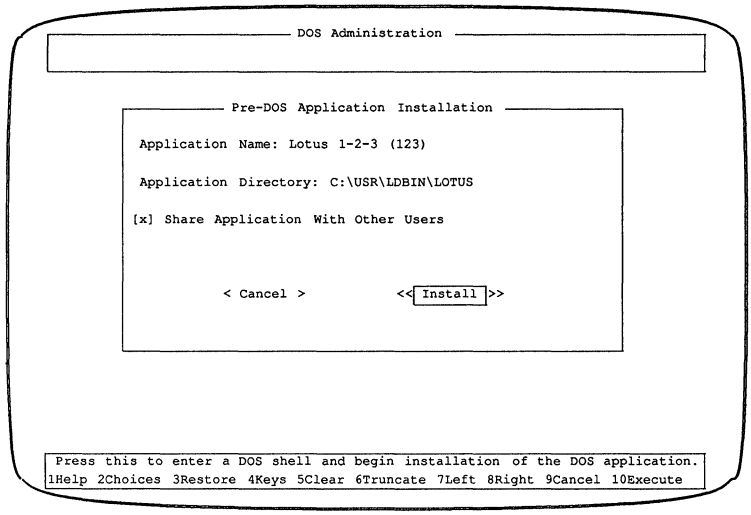


When you check the "Share Applications" box, **dosadmin** sets permission modes so all users can run the application.

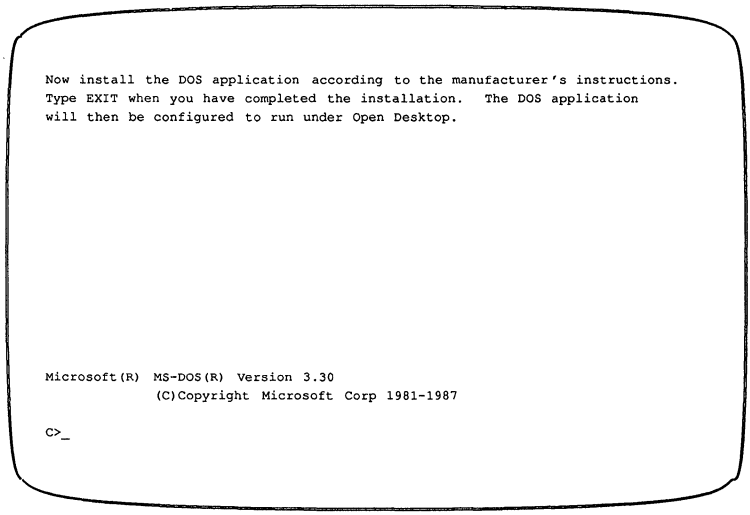
10. Move to the "< Cancel >" field by pressing the **Tab** key. When "< Cancel >" is highlighted, you can press **Enter** to cancel the installation procedure and return to the top-level **dosadmin** menu.

    If you choose to continue the installation, check your entries for accuracy. If you want to make any changes, you can return to any field by pressing the **Tab** key repeatedly, and then type in your corrections.

11. When you have confirmed your entries, press the **Tab** key until "<< Install >>" is highlighted like this:



12. Press **Enter** to save your choices and start the next part of the installation process. Your display is now similar to this:



13. Continue the installation procedure by following the application manufacturer's instructions for installation on a personal computer fixed disk. Your current working directory and drive are those you specified in the "Application Directory" field of the Pre-DOS Application Installation menu. If the manufacturer's instructions require you to identify the drive or directory in which you are installing the application, you should normally enter the same values you used in the Pre-DOS Application Installation menu. However, if you change your mind about the drive or directory, you can correct the **dosadmin** database later.

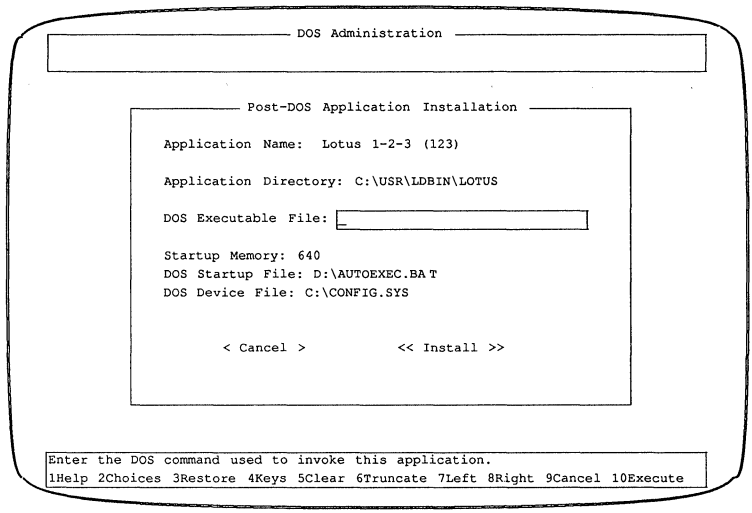
14. When you have completed installation according to the manufacturer's instructions, type:

    C> **exit**

    WARNING: Be sure you type **exit** and not **quit** when you have finished the manufacturer's recommended installation procedure. If you type **quit**, you have to start the installation procedure over from the beginning.[1]

[1] However, if you have finished installing the application before accidentally typing **quit**, you can skip step 13 when you redo the installation procedure. Since the application is already installed, there is no need to install it again.

The Post-DOS Application Installation menu appears, as follows:



15. The "DOS Executable File" field is highlighted. Type in the command you use when you run the program. For example, type **123** if this is the command you use to run Lotus 1-2-3.

    If you use more than one command with this application, see "Configuring DOS Applications for Use from the UNIX Shell" later in this chapter.

16. Check all fields of the display for accuracy. Most applications will run correctly without further changes to this display, but you should now correct or change the displayed values if necessary. To make changes, press the **Tab** key repeatedly until the field you want to change is highlighted. Then type in the correct information.

Following are points you should consider as you review the values for each field:

- **Application Name:** The displayed name is the name you entered during the pre-installation phase. The **dosadmin** database associates the application name shown here with the command shown in the ''DOS Executable File'' field.

- **Application Directory:** This field shows the drive and directory you entered during the pre-installation phase. If you installed the application on a different drive or in a different directory for any reason, you should correct this field now. The directory name displayed must be that of the directory containing the executable DOS file (for example, 123.COM).

- **DOS Executable File:** This is the command you type when using the application. The **dosadmin** database associates the command name shown here with the application name shown in the ''Application Name'' field.

- **Startup Memory:** This is the amount of memory (in Kbytes) reserved for this application when you run it from the UNIX shell. The value displayed is 640 Kbytes, unless factory defaults have been changed. All applications run correctly with 640 Kbytes of memory, but not all applications need this much memory. If your application runs correctly with less memory, you can increase ODT-DOS efficiency by specifying a smaller value than 640. Refer to the application manufacturer's instructions for the recommended memory, and enter it here.

- **DOS Startup File:** The file named in this field (D:\AUTOEXEC.BAT by default) is executed automatically whenever you run the application from the UNIX shell. When you are logged in as root, D:\AUTOEXEC.BAT is the root directory AUTOEXEC.BAT in the shared DOS/UNIX filesystem. That is, D:\AUTOEXEC.BAT is just another name for C:\AUTOEXEC.BAT or **/autoexec.bat**. If you have created or added to the D:\AUTOEXEC.BAT file while installing this application, it will be executed as expected when you run the application.

Users' home directory AUTOEXEC.BAT files run automatically in addition to D:\AUTOEXEC.BAT when they execute the DOS application from the UNIX shell as long as the default D:\AUTOEXEC.BAT file is displayed in the "DOS Startup File" field. If you want only the root directory AUTOEXEC.BAT to run when users invoke this application, replace the displayed D:\AUTOEXEC.BAT with C:\AUTOEXEC.BAT.

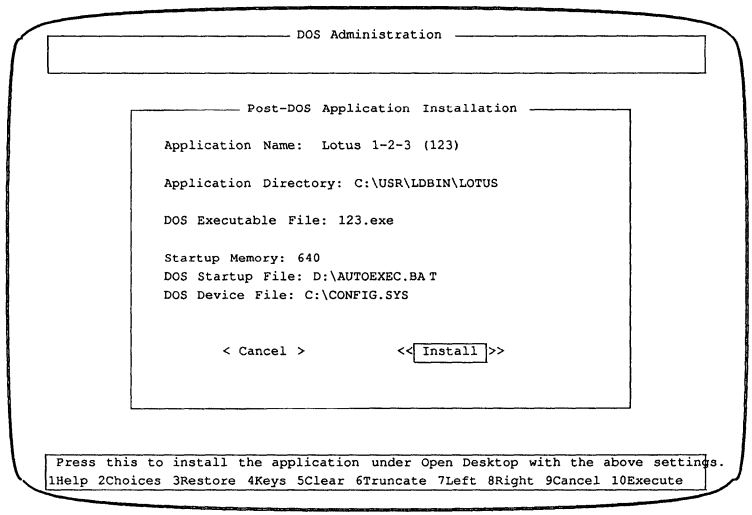
If you have created a file other than D:\AUTOEXEC.BAT that should run every time you use this application, enter the name of the file in this field. For example, if your application is supposed to run the file C:\USR\LDBIN\APPL.BAT every time you invoke it, type in this field:

```
C:\usr\ldbin\appl.bat
```

If you do not want any AUTOEXEC.BAT file to run automatically when you use this application, erase the field by pressing the **F5** key.

- **DOS Device File:** This field lists the device configuration file interpreted by DOS when you run your application from the UNIX shell. The system default for this field is a single file—C:\CONFIG.SYS—when you install a public application. You can accept or change this value as appropriate for the application you are installing. Note that, even though **dosadmin** displays only the system default device file, when users run the application, two device files, C:\CONFIG.SYS and their home-directory CONFIG.SYS, are interpreted if they both exist. (Home directory CONFIG.SYS files do not exist until users create them.)

17. When the display shows correct information, press the **Tab** key until "Install" is highlighted. The display should resemble this:



18. Press **Enter** to complete the installation procedure. An "In Progress" message appears while **dosadmin** configures your application. Then the top-level **dosadmin** menu returns.

    If you want to install another application, press **Enter** to redisplay the Install Applications menu. Otherwise, to return to your system prompt, press the **Left** or **Right Arrow** key (← or →) until "Exit" is highlighted, and press **Enter**.

Defining the Search Path

After you install a DOS application, you may want to update the search path if you have installed the application in a directory that is not already in users' search paths. You can update the search path for all users by changing the **PATH** definition in **/etc/default/login**.

The UNIX search path applies by default both at the UNIX shell and in the DOS environment.

Installing Personal DOS Applications

The procedures for installing personal DOS applications are almost identical to the procedures described in the previous section for public DOS applications. To install personal applications, refer to the instructions in the previous section, "Installing Public DOS Applications," and note the following differences:

- **Step 1:** Log in using your normal login name. Your prompt is **$** instead of **#**.

- **Steps 6 and 7:** The default directory for installing personal DOS applications is D:\ (your home directory). Normally, you should not change this field to show a different drive from drive D:. You may, however, want to install your DOS application in a subdirectory subordinate to your home directory. You can specify a subdirectory by filling in this field appropriately.

- **Step 9:** You are not required to share the application with other users. If you want it to be executable by others, press the **Space** or **Enter** key to check the "Share Application With Other Users" box. Otherwise, just press **Tab** and continue with the rest of the application installation procedures.

- **Step 16:** When you install personal DOS applications, the "DOS Startup File" field displays D:\AUTOEXEC.BAT by default, just as it does when you install public DOS applications. When you are not logged in as root however, D:\AUTOEXEC.BAT is your home directory AUTOEXEC.BAT rather than the root AUTOEXEC.BAT. If the "DOS Startup File" field displays "D:\AUTOEXEC.BAT," DOS interprets both the root and your home directory AUTOEXEC.BAT files (if they both exist) when you run this application from the UNIX shell. If you want a different startup file to be interpreted when you run this application, you can change this field as described in step 16 under "Installing Public DOS Applications."

  When you install personal DOS applications, the "DOS Device File" field by default shows two files: C:\CONFIG.SYS and D:\CONFIG.SYS. These files are the system default CONFIG.SYS file in the root and your home directory CONFIG.SYS file. DOS interprets both files, if they are both listed and both exist, when you run this application from the UNIX shell. (If no filename is displayed, then no device configuration file is interpreted when you run this application.) If your application requires device configuration information from a CONFIG.SYS file, it must be listed here.

Even if your application does not require a CONFIG.SYS file, you should normally not delete the displayed default filenames. If your system hardware changes in the future, one or more of the listed files may contain configuration information required when you run any DOS program.

Defining the Search Path

After you have installed a personal DOS application, you may want to update your search path. You can change the UNIX search path that is defined whenever you log in by modifying your home directory **.profile** file. The UNIX search path applies by default in the DOS environment also.

You can also use standard DOS methods to define your search path in the DOS environment.

Configuring DOS Applications for Use from the UNIX Shell

You can use **dosadmin** to configure DOS applications that are already installed so they can be used from the UNIX shell. This procedure is useful under the following circumstances:

- You previously installed a DOS application that has never been configured for use from the UNIX shell.
- You previously installed an application such as Lotus 1-2-3 that has more than one executable file and has configured only one file for use from the UNIX shell.
- You want to change the characteristics previously assigned to an application with **dosadmin**.

To configure an already-installed application using **dosadmin**, you simply follow all the steps listed under "Installing Public DOS Applications" with one exception: do not complete step 13, installing the application according to manufacturer's instructions, since the application is already installed. Instead, just type **exit** at the DOS prompt (step 14) and continue with the Post-DOS Application Installation menu.

During this procedure, **dosadmin** displays the application's current values for startup memory, startup file, and device files. You can change these values before exiting **dosadmin** if you wish.

When you exit **dosadmin**, the application is added to the **dosadmin** database, and you can access the application using the **dosadmin** List Applications, Remove Applications, and Tailor Applications menus.

How dosadmin Configures Applications

dosadmin automatically accomplishes the following operations when you use it to install DOS applications:

- Adds the name of the application to a database of installed applications. The database is used by other **dosadmin** functions, including the List Applications, Remove Applications, and Tailor Applications menus.
- Links DOS executable files to UNIX files without **.bat**, **.com**, or **.exe** extensions.
- Sets UNIX permission modes so the application can be used from both the DOS environment and the UNIX shell.
- Adds appropriate **dos** options. (Refer to *Using ODT-DOS* for further information on **dos** options.)

NOTE: When you install DOS applications using **dosadmin**, **dosadmin** creates a file named **sdfile** that stores information about the applications. If you are logged in as a user and are installing personal applications in a directory you own, **sdfile** is stored in your home directory. If you are the system administrator logged in as root, **sdfile** is stored in **/usr/lib/merge**. You need not be concerned with the contents of **sdfile**, but be sure you do not accidentally delete it.

Installing Copy-Protected DOS Applications

A few DOS applications are designed by the manufacturer to prevent installation on more than one machine or execution by more than one user at a time. You can use these applications with Open Desktop, but the installation procedures differ according to the type of copy protection used.

Many different copy-protection methods are in use, and it is impossible to recommend installation procedures that apply universally. There are, however, two common classes of copy-protected applications for which we can supply general recommendations that apply in most cases. These classes are:

- applications that use "key disks."
- applications that use special installation utilities.

Procedures for installing applications in each of these classes are described in the following sections.

Using a Key Disk

An application that uses a key disk allows you to install part of the application on the system fixed disk but requires you to insert a protected diskette in the diskette drive whenever you use the application.

Such applications generally require no special installation procedures to work on Open Desktop. You can install them by running a DOS environment and following the manufacturer's instructions or the instructions in the previous section, "Installing DOS Applications Using **dosadmin**." When you run an application that requires a key disk, you use the key disk just as you would on a conventional personal computer running DOS.

Using Special Installation Procedures

Another class of copy-protected applications uses special installation (and sometimes also removal) procedures to prevent illegal copying of the application. These applications typically include a special installation utility (often invoked with a command such as "install"). They are intended to be installed in their entirety on the system fixed disk so that no key disk is required. You can install such applications on Open Desktop but, because different copy-protection methods are in use, you may have to try more than one installation procedure. The procedures, listed in the order you should try them, are:

- If you do not intend to use the application from the UNIX shell, run a DOS environment and install the application according to the manufacturer's instructions.
- Install the application according to the standard instructions under "Installing DOS Applications Using **dosadmin**," earlier in this chapter.

- Install the application on DOS drive E: (the DOS partition). This procedure is described in the next section, "Installing DOS Applications on the DOS partition."

- Shut down Open Desktop, boot DOS, and install the application on the DOS partition. This procedure is also described in the next section.

Installing DOS Applications on the DOS Partition

This section describes two methods of installing DOS applications on the DOS partition. These procedures are particularly useful when you install copy-protected applications that cannot be installed using **dosadmin**.

Installing DOS Applications under Open Desktop

This procedure is required for a few copy-protected DOS applications that must be installed on an actual DOS filesystem rather than on the shared DOS/UNIX filesystem. This installation method works only if you have a DOS partition on Open Desktop. The DOS partition can be a physical partition (that can be booted and run independently of Open Desktop), or it can be a virtual partition. If your system does not have a DOS partition, you can create one as described in Chapter 2 of this guide.

NOTE: These instructions assume you are installing an application on the physical DOS partition (drive E:). If you are installing on a virtual partition, you must first attach the partition to your DOS process as described in "Administering Virtual DOS Partitions and Virtual Floppy Disks" in Chapter 2 of this guide.

1. Log in to Open Desktop. If you are installing an application in a virtual partition that you own, just log in as you usually do. If you plan to install an application in the physical DOS partition, you must log in as a user who has permission to read and write the physical DOS partition. (By default, all users have read and write permission.)

2. Start the DOS environment by typing (at the UNIX prompt):

   $ **dos**

   Your prompt changes to the DOS prompt **C>**.

3. Change your working drive to drive E: by typing:

   C> **e:**

4. Make sure you are working in the directory in which you want to install the application. If necessary, use the DOS CD (change directory) command to move to the desired directory.

5. Install the DOS application according to the manufacturer's instructions. For example, if the application has an installation program named INSTALL, invoke it by entering (at your DOS prompt):

   E> **install**

   (If the manufacturer provides no other installation instructions, use the DOS COPY command to transfer all files from drive A: to drive E:.)

The application is now usable from DOS drive E:.

Installing Applications under "Raw" DOS

Use this procedure for the few copy-protected DOS applications with special timing requirements that make them impossible to install while ODT-DOS is running.

NOTE: This method works only if your system has a physical DOS partition that can run DOS as a stand-alone operating system.

Since you boot your system under DOS during this procedure, you must have one of the following:

- A bootable DOS system diskette. If you do not already have such a diskette, you can create one by inserting a diskette into drive A: and typing:

  $ **dos format a: -s**

- A bootable DOS partition. (It is bootable if you formatted it using the FORMAT /S option.)

Follow these procedures:

1. Shut down the UNIX system according to the instructions in the *Administering ODT-OS*.

2. Reboot the system from a bootable DOS diskette or from the DOS partition.

3. Follow the manufacturer's instructions to install the application on the DOS partition. The DOS partition is accessible as drive C: when you are running raw DOS.[1]

4. When you are finished installing the application, reboot the UNIX system.

The application should now be usable like any other application on the DOS partition. When ODT-DOS is running, the DOS partition is drive E:.

Removing DOS Applications

Following is an outline of the procedures you follow to remove installed DOS applications from the Open Desktop fixed disk. Each of these steps is described in the following sections.

- Remove the application according to the manufacturer's instructions.
- Remove UNIX links to DOS executable files.
- Remove the **dosadmin** database entry for the application, if there is one.
- Redefine the search path, if necessary.

[1] A few DOS applications must be installed and run on the same drive. Since you access the DOS partition as drive E: when ODT-DOS is running and as drive C: when you have booted raw DOS, you should not install these applications directly on drive C:. While your system is running raw DOS, before installing one of these applications, use the DOS SUBST command as follows:

```
C> subst e: c:\
```

The DOS partition is then accessible as drive E:. Continue with the installation procedure, referring to the DOS partition as drive E:.

Removing the Application from the Fixed Disk

Remove a DOS application from the Open Desktop fixed disk by following the manufacturer's instructions. If no instructions are provided, it is usually safe to remove the application's files using the UNIX **rm** or DOS DEL command.

Note that some copy-protected applications require special removal procedures. If you do not follow the required procedure, you may not be able to reinstall the application at a future time.

Deleting UNIX Links

If the DOS application you have removed was configured for execution from the UNIX shell, there are UNIX files that were linked to DOS executable files. (For example, **ws** would be linked to WS.COM.) When you remove DOS executable files (files with extensions ending in .COM, .EXE, or .BAT), the corresponding UNIX links are not automatically removed. When you remove a DOS application, you should also delete any UNIX links to the DOS files you are removing. To delete these links, you simply use the UNIX **rm** or DOS DEL command:

```
$ rm ws
```

Removing the dosadmin Database Entry

If you used **dosadmin** to install or configure your application when you installed it, there is an entry for the application in the **dosadmin** database. When you remove an application, you should remove its database entry. To remove a **dosadmin** database entry for an application, proceed as follows:

1. From either the UNIX shell or the DOS environment, type:

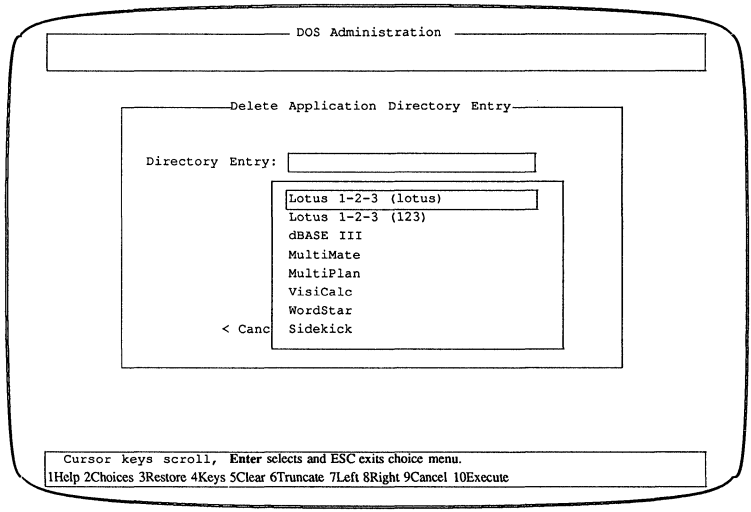
   ```
   dosadmin
   ```

   The main **dosadmin** menu screen is displayed.

2. If the "Applications" menu is not already highlighted, use the **Left** or **Right Arrow** key (← or →) to highlight it.

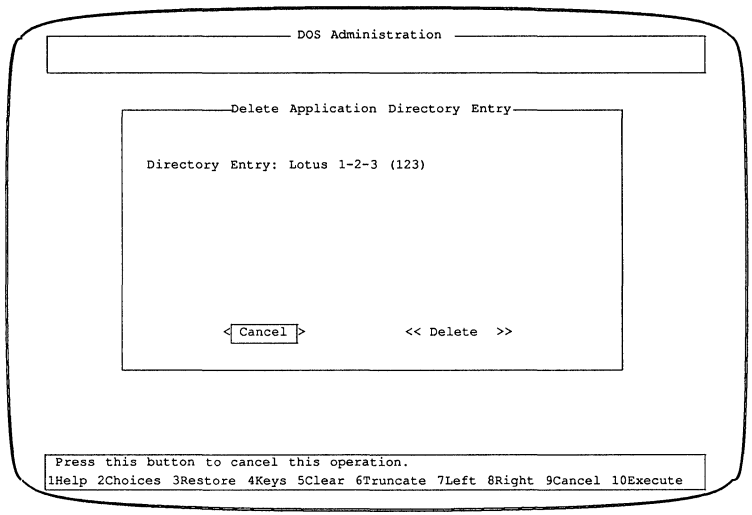
3. Press the **Down Arrow** key (↓) to move the highlighted field to "Remove."

4. Press **Enter** (↵) to display the "Remove Application" menu. The menu displays a list of the applications currently installed in the **dosadmin** database, like this:



5. Select the application you want to remove by pressing the **Down Arrow** key (↓) until the application name is highlighted. If there are more installed applications than there is room to display, you can press the **Down Arrow** key when you reach the bottom of the window to scroll the list and display additional names. To move up in the list, use the **Up Arrow** key (↑), which also scrolls when you reach the top of the window.

6. When the application you want to remove is highlighted, press **Enter**. The application you have chosen is displayed in the "Directory Entry" field, like this:



7. If you choose to cancel the operation and leave the application in the **dosadmin** database, press **Enter** while "< Cancel >" is highlighted. Otherwise, to remove the application, press **Tab** (⇆) to move the highlighted field to "<< Delete >>."

8. Press **Enter**. The application is then removed from the **dosadmin** database, and the Applications menu reappears on your screen.

9. Press the **Left** or **Right Arrow** key (← or →) to move the highlighted field at the top of the menu to "Exit."

10. Press **Enter** to exit **dosadmin** and return to your system prompt.

Redefining the Search Path

After you have removed a DOS application, you may want to redefine your search path. Depending on whether the application you removed was a public or a personal application and on how you define your search path, you might want to modify the **PATH** definitions in **/etc/default/login**, your home directory **.profile** file, or an AUTOEXEC.BAT file.

Administering ODT-DATA

ODT-DATA is based on technology developed by **INGRES CORPORATION**, and includes the following **INGRES** components:

- INGRES/DBMS and SQL Terminal Monitor
- INGRES/User Interfaces
  - Query-by-Forms
  - Report-by-Forms
  - Report Writer
  - Menu
  - Forms Runtime Systems and VIFRED
- INGRES/NET with TCP/IP Support
- INGRES/WindowView
- INGRES/ESQL Preprocessor for C

12/21/89-1.0.0D
Processed: Wed Dec 20 11:08:19 PST 1989

Contents

Chapter 1
Introduction

ODT-DATA is a relational database management and application development system, based on technology developed for INGRES by Relational Technology, Inc.

This book describes the procedures for installing ODT-DATA and authorizing user access to specific databases. It also provides information for the ODT-DATA system administrator.

The ODT-DATA system administrator is the individual responsible for the ODT-DATA installation and the authorization of user access to ODT-DATA. The ODT-DATA system administrator is also responsible for installing ODT-DATA updates and new releases, but does not necessarily have responsibility for databases maintained under the ODT-DATA system. A database administrator (DBA) is designated for each database. The DBA is usually the user who creates a database. The DBA is responsible for the creation of shared tables and authorization of user access to shared tables. The DBA is also responsible for backup and recovery of the database.

Important Note

ODT-DATA installation requires access as "root" on UNIX®. Traditionally, access to "root" privileges presumes a knowledge of the UNIX operating system. This publication assumes that anyone installing ODT-DATA has such a knowledge of UNIX.

If you possess the "root" password and do not have minimal knowledge of UNIX refer to the UNIX sections of the Open Desktop™ *User's Guide* and *Administrator's Guide*. If you are unfamiliar with UNIX, be sure that someone with UNIX expertise can assist you with the installation of ODT-DATA.

Read this publication before proceeding.

Introduction to Release 6

The ODT-DATA DBMS server architecture provides access to the DBMS through a shared process. With the server architecture comes a logging and locking facility that is configured during the ODT-DATA installation. The DBMS logging facility is implemented by two processes. The components of the logging system include: a process to handle online recovery and a process to archive journaled data. An installation-wide logging file keeps track of all ODT-DATA transactions and calls within the DBMS server. The logging facility logs ODT-DATA transactions, manages the logging file, and insures that log records are accessible to the recovery and archiver routines.

The **recovery** process (**dmfrcp**) handles online recovery from system failures and inconsistencies generated by user actions. Consistency points are written into the logging file to allow online recovery when a problem is detected. This permits other users to continue working in the database while the inconsistency is corrected.

The **archiver** process (**dmfacp**) removes completed transactions from the logging file and writes them to the corresponding journal files for the particular database. This process sleeps until sufficient portions of the logging file are ready to be archived. The ODT-DATA system administrator can tune the values that define the logging system.

The recovery and archiver processes are daemon processes started at boot time. They are available on the system at all times and reduce the startup times required to begin an ODT-DATA session. They also reduce the resource requirements, since sessions can use a single process to service DBMS requests, the server.

Organization of This Document

This book contains the following items:

- Chapter 1 introduces this book.
- Chapter 2 describes various installation tools:
  - the ODT-DATA installation utility (**iibuild**),
  - the ODT-DATA installation start up utility (**iistartup**), and
  - The ODT-DATA installation shut down utility (**shutserver**).

- Chapter 3 discusses various system requirements and configuration decisions.
- Chapter 4 describes how to install ODT-DATA
- Chapter 5 discusses maintenance utilities, including:
  - server monitoring and control utility (**iimonitor**),
  - shared memory and semaphores report utility (**csreport**),
  - locking facility report (**lockstat**), and
  - logging facility report (**logstat**).
- Chapter 6 discusses miscellaneous topics in installation:
  - DBMS server creation utility,
  - the logging and locking facility, and
  - ODT-DATA Release 6 shutdown procedure.
- Chapter 7 discusses troubleshooting ODT-DATA with log files.
- Appendix A discusses ODT-DATA startup files.
- Appendix B discusses authorizing user access to ODT-DATA and databases.
- Appendix C discusses recovery of the ODT-DATA master database.
- Appendix D discusses ODT-DATA environment variables.
- Appendix E discusses running ODT-DATA under Network File System (NFS™).

Associated Publications

The table below lists all the ODT-DATA books available with each Open Desktop product:

Open Desktop:
■ *Administering ODT-DATA*
■ *Using ODT-DATA*
Open Desktop Development System:
■ *ODT-DATA Embedded SQL User's Guide*
■ *ODT-DATA Embedded Open SQL Forms Reference Manual*
■ *ODT-DATA Open SQL Reference Manual*
■ *ODT-DATA Embedded SQL Companion Guide for C*
Open Desktop Server Upgrade:
■ *Administering an ODT-DATA NET Server*
Open Desktop Optional Documentation:
■ *Using ODT-DATA Through Forms and Menus*
■ *ODT-DATA Report-Writer Reference Manual*
■ *ODT-DATA SQL Reference Manual*

Chapter 2
Overview of Installation Tools

This chapter provides an overview of the installation tools. The following tools are located in the *$II_SYSTEM/utility* directory and can only be executed by the "ingres" user. Only the "ingres" user has access to these tools.

ODT-DATA Installation Utility—iibuild

The ODT-DATA Release 6 installation utility is **iibuild**. This script is run by the "ingres" user during the ODT-DATA installation procedure described in Chapter 4. The **iibuild** utility performs the following tasks during installation:

- ensures that the ODT-DATA environment is set up for installation
- checks that there are sufficient system resources for installation
- sets up the ODT-DATA installation, including ownership and permissions
- installs system logging and locking resources
- installs the ODT-DATA log file
- installs and starts logging, locking, and the DBMS server
- creates the database database (**iidbdb**)

ODT-DATA Installation and DBMS Server Start Up

You can use **iistartup** to start up a single server or an installation. The **iistartup** utility is used by the **iibuild** script to start up your ODT-DATA installation during the ODT-DATA installation procedure.

Depending upon the option used to invoke the utility, **iistartup** will either start up an installation automatically or prompt you through your choice of the following tasks:

- install shared memory resources
- configure and create a log file
- configure logging and locking parameters
- initialize a log file
- start up the recovery process
- start up the archiver process
- configure the DBMS options and start up the server

If you choose a configuration option, you are prompted for parameters. Refer to Chapter 6 for configuration parameters and **iistartup** syntax.

The Installation Shut Down Utility—shutserver

shutserver is a utility that can be used to shutdown individual servers or the entire installation. **shutserver** will prompt for the following options:

- Bring down server process(es) designated by communication address
- Reconfigure the server(s)
- Stop the installation's archiver and recovery processes
- Change the logging and locking facility parameters
- Deinstall shared memory

Refer to Chapter 6 for **shutserver** syntax and emergency shutdown information.

Chapter 3
Configuration Decisions

This chapter outlines the system requirements to configure and install ODT-DATA and gives suggestions for avoiding problems with your ODT-DATA installation.

Configuration Requirements

This section discusses the requirements for disk space, memory, semaphores, and swap space, as well as the required ODT-DATA environment variables.

Disk Space Requirements

Approximately 32 Mbytes of disk space are required to load your ODT-DATA installation media. In addition, a minimum of 4,096 Kbytes are required for your log file. You must also estimate the space needed for databases, journals and checkpoints.

Memory Requirements

To run ODT-DATA, you must configure your UNIX kernel with sufficient shared memory resources to support ODT-DATA installation, locking and server requirements. Your ODT-DATA installation requires 208,000 bytes of shared memory; 8,192 bytes for a system segment and a minimum of 204,800 bytes for a lock segment. In addition, each DBMS server requires a shared memory segment. The size is calculated as '16,384 + (8,708 * max_connected_sessions)'. The value for "max_connected_sessions" is the same used for the server parameter of that name. See Chapter 6 for server parameter information.

For example, to configure shared memory for an installation with a single server, using the default maximum of 25 connected sessions.

Allow 8,192 bytes of shared memory for the installation segment.

Allow a minimum of 204,800 bytes of shared memory for the locking segment.

Calculate the shared memory for the server segment::

16,384 + (8,708 * 25) = 234,084 bytes of shared memory

Add:

8,192 + 204,800 + 234,084 = 447,076

The total shared memory requirement for this installation is a minimum of 447,076 bytes.

The minimum shared memory configuration possible is 238,084 bytes. This would be single server installation, configured for a maximum of one connected session.

Semaphore Requirements

Your ODT-DATA installation requires a semaphore set of 21 semaphores (5 + maximum_number_servers, which is a hard-wired limit of 16). In addition, each server requires a separate set of 10 semaphores. A minimum system configuration requires 31 semaphores.

Swap Space Requirements

A minimum of 16 megabytes of swap space is recommended for an ODT-DATA installation with one DBMS server. You need to add an additional 3 to 5 megabytes for each additional DBMS server, depending on the configuration of maximum connected sessions.

Environment Variables

The following environment variables are set during the ODT-DATA installation procedure. These are set to defaults by **iibuild** during the install process. In the standard installation the defaults are set for you. If you want a manual installation you need to decide what to set them to before running **iibuild**.

II_SYSTEM

In the standard installation the default location for your ODT-DATA installation is set for you. The environment variable II_SYSTEM is hard coded to */usr*. Before you begin a manual installation you need to determine the location for your ODT-DATA installation, so the environment variable II_SYSTEM is used by ODT-DATA programs to locate the DBMS installation. Before you begin the installation procedure, this variable must be set in the "ingres" user's environment to the full path name of the ODT-DATA installation directory.

Because II_SYSTEM is not set in the ODT-DATA symbol table, it must be set by users in their environment before they can run ODT-DATA.

Use the $II_SYSTEM directory only for ODT-DATA administration. Do not create user files or directories in the $II_SYSTEM directory or its subdirectories.

II_INSTALLATION

Your ODT-DATA installation, II_INSTALLATION, is identified by a unique two-letter identification code, which you select. Your code must be two alphanumeric characters such as "xx" or "x4", and the first character must be a letter.

II_LOG_FILE (Logging File Location)

ODT-DATA Release 6 uses an installation-wide logging file. This file handles all of the installation's concurrent ODT-DATA transactions. The logging facility writes pending transactions to the logging file, and the archiver facility moves completed transactions to the journal files when necessary. The full path name of the log file is *$II_LOG_FILE/ingres/log/ingres_log*.

The default value for II_LOG_FILE is the value of II_SYSTEM. Sites must determine the best place for this file to reside. Do not place the logging file on an I/O bound disk. Data acquisition times of the recovery and archiver routines will increase, slowing down all users on the ODT-DATA installation.

The size of the logging file is another important factor. The logging file must be large enough to handle all concurrent transactions without reaching saturation. It is designed as a circular file that wraps when the physical end-of-file is reached. If the logging file reaches the force-abort-limit, the oldest transactions are backed out dynamically. If it is not successful in freeing enough space and the log-full-limit is reached, the transaction system stops and backs out the oldest transactions. This could have severe performance implications and should be avoided.

The Raw Log File Option

The ODT-DATA log file can be configured as a "raw device." Utilizing a "raw device" for the log improves performance by:

- writing directly to disk, bypassing the UNIX cache
- writing larger disk blocks

If you plan to use a raw device, you must complete the following steps before running the **iibuild** installation script:

- The raw log device must be created as a character special device.
- The raw device cannot contain a filesystem.
- The device must be owned by the "ingres" user.
- The "root" user must run the *$II_SYSTEM/utility/mkrawlog* script to link the *ingres_log* file to the designated raw device file.

II_DATABASE

The II_DATABASE variable is set during ODT-DATA installation and may not be changed later, even during updates, so select your location carefully. The II_DATABASE environment variable points to the default location for database files. This environment variable also determines the location of the ODT-DATA master database, the **iidbdb**. Any database except the **iidbdb** may be created on alternate locations and moved later.

On multi-disk systems, II_DATABASE should not be set to a directory on an I/O-bound system disk because ODT-DATA database scans should not compete with system operations (such as system calls) for I/Os. Do not set II_DATABASE to a full filesystem, or to one that will become full as databases are added. Full disks become fragmented, and disk performance degrades.

II_CHECKPOINT and II_JOURNAL

The II_CHECKPOINT and II_JOURNAL variables are set during ODT-DATA installation and cannot be changed later, even during updates. II_CHECKPOINT is the environment variable set to the location iicheckpoint , where ODT-DATA checkpoints reside. II_JOURNAL is set to the location iijournal, where ODT-DATA journal files reside. Checkpoints are static backups of a database, while journals are dynamic records of changes made to a database since the last checkpoint. A checkpoint provides for recovery up to the time the checkpoint was taken. Checkpoints and journals provide for complete recovery up to the time of failure.

On single-disk systems, checkpointing to disk provides little safety. Disks usually crash in an all-or-nothing fashion. On single disk systems, checkpointing to magnetic tape is recommended. Storing data and backups on the same device provides little insurance from a disk failure. You can put journals and checkpoints on the same device. Journals are used in recovery only if the associated checkpoint is also available.

General Suggestions for Avoiding Problems

The *$II_SYSTEM/ingres* directory will usually be the home directory of the ODT-DATA system administrator, the "ingres" user. This account should be used only for ODT-DATA administration. No user files or directories should be placed in the $II_SYSTEM directory or its subdirectories.

The files in the $II_SYSTEM directory and subdirectories are critical to ODT-DATA. Deleting or changing any of these files may corrupt your installation.

ODT-DATA uses operating system permissions to protect your data. Never alter the permissions of any ODT-DATA file, directory, or subdirectory.

Chapter 4
Installing ODT-DATA

Read the preceding chapters before installing ODT-DATA. Installation requires some knowledge of UNIX. ODT-OS and TCP/IP must be installed before you can install ODT-DATA. After the ODT-DATA software is installed, it must be initialized. TCP/IP must be running before you initialize ODT-DATA. To initialize ODT-DATA automatically, log in as *ingres* and respond affirmatively to the initialization prompt.

Therefore, before using ODT-DATA, you must do the following:

1. Installl ODT-OS;
2. Install ODT-NET TCP/IP;
3. Install ODT-DATA;
4. Activate TCP/IP;
5. Initialize ODT-DATA by logging in to the system as *ingres* and responding affirmatively to the initialization prompt.

The next section presents the manual initialization, in which you must answer a series of prompts.

Manual Initialization

The automatic ODT-DATA initialization is accomplished by logging in as user **ingres** and responding **y** at the prompt for initialization. This automatic initialization uses the defaults described in the table below and you are not prompted for configuration information such as log file size and number of log buffers. To initialize ODT-DATA manually, you must respond **n** to the prompt for initialization. You should then run the **iibuild** command manually to customize ODT-DATA as desired. You can accept the default entries or enter your own values.

ODT-DATA Default Initialization Parameters	
Parameter	**Value**
Log file size 32 Kbyte blocks	128
Number of log buffers	4
Maximum number of databases in logging system	32
Maximum number of transactions in logging system	32
Block size of the log file	4 Kbytes
Log-full-limit	95%
Percent of log file for each consistency point	5%
Maximum consistency point interval for invoking archiver	19%
Force-abort-limit	80%
Size of locks hash table	63
Size of resources hash table	63
Maximum number of locks in locking system	2000
Maximum number of lock lists in locking system	128
Maximum number of locks allowed per transaction	150
Maximum number of server connections	50
Maximum number of open cursors per session	16
Maximum number of open databases per server	25
Per session stack size in bytes	32768

Initialization Checklist

You need to collect all the following information, before you begin the ODT-DATA initialization procedure. Please fill in the blanks on this checklist.

Does your UNIX system configuration meet the minimum ODT-DATA installation requirements?

238084 bytes or more of shared memory (y/n)_____

31 or more semaphores (y/n)_____

16 Megabytes or more of swap space (y/n) _____

If any of your answers are *no*, read Chapter 3 before continuing.

- *Your media device name is :*_____
- *Your two-letter ODT-DATA installation code is :*_____
- Read the configuration discussion in Chapter 3, and choose default locations for your databases, checkpoints, journals, and logging file.

  *For your databases, set II_DATABASE to :*_____

  *For your checkpoints, set II_CHECKPOINTS to :*_____

  *For your journals, set II_JOURNAL to :*_____

  *For your the logging file, set II_LOG_FILE to :*_____

- *Do you want the log file for this installation to be a raw device?* (y/n)_____

See Chapter 3 for instructions on creating a raw log device and for log file information.

Read the discussion in Chapter 6 of this book for an explanation of logging and locking facility configuration and primary DBMS server information.

Logging Parameters

*Log file size in 32Kb blocks :*_____

*Number of log buffers in memory :*_____

*Maximum number of databases :*_____

*Maximum number of transactions :*_____

*Block size of the log file :*_____

*Log-full-limit in percentage :*_____

*Percentage of log file for each consistency point :*_____

*Maximum number of consistency points written to invoke the archiver :*_____

*Maximum consistency point interval for invoking archiver :*_____

*Force-abort-limit in percentage :*_____

Locking Parameters

*Size of the locks hash table :*_____

*Size of the resources hash table :*_____

*Maximum number of locks in locking system :*_____

*Maximum number of lock lists in locking system :*_____

Maximum number of locks allowed per transaction : _____

See Chapter 6 of this book for an explanation of the options used in the startup of the DBMS server.

*Maximum number of server connections allowed :*_____

*Maximum number of open cursors per session :*_____

*Maximum number of open databases per server :*_____

*The per session stack size in bytes :*_____

Do you want this to be a "sole server" (required for fast commit)? (*y*/*n*)_____

Do you want the write_behind option set? (*y*/*n*)_____

Do you want to create a dblist for this server? (*y*/*n*)_____

Choose the UNIX editor to be used by ODT-DATA. The default is the current setting of the UNIX variable $EDITOR

*Set ING_EDIT to :*_____

Do you have ODT-DATA NET? (*y*/*n*)_____

Chapter 5
Maintenance Utilities

Use the maintenance and report utilities listed in this chapter to monitor and administer your ODT-DATA installation.

The Server (iidbms) Maintenance Utility—iimonitor

Database access using ODT-DATA Release 6 is controlled by the DBMS server. Use the **iimonitor** utility to examine the status of a server and the connected sessions. Use the utility to control the server's execution, including shutting down sessions and DBMS server. The administrative options such as stopping the server are restricted to an ODT-DATA superuser. This utility will:

- Display DBMS server information
- Display active sessions for the DBMS server
- Stop the DBMS server
- Display session information
- Disconnect a session
- Suspend a session

The II_DBMS_SERVER Environment Variable

Set the II_DBMS_SERVER environment variable to the communications address of the server you want to monitor with the **iimonitor** utility. To examine the global value of $II_DBMS_SERVER, type:

```
$ ingprenv | grep II_DBMS_SERVER
```

If a new server is created without specifying **-nopublic**, II_DBMS_SERVER is set to the new server's address. Connection to the old server is not possible without explicitly setting II_DBMS_SERVER to the old server's address. For example, to examine an old server with the communications address "1367", do the following:

C shell example:

```
%  setenv  II_DBMS_SERVER  1367
```

Bourne shell example:

```
$  II_DBMS_SERVER=1367
$  export  II_DBMS_SERVER
```

The iimonitor Utility

The **iimonitor** utility allows you to examine the status of a server and the sessions connected to it. Administrative options, like stopping the server, are restricted to the ODT-DATA superuser. The utility can be used to control the server's execution, including shutting down sessions or the server itself. At the operating system prompt, to start the **iimonitor** utility, type:

```
$  iimonitor <server name>
```

At the "IIMONITOR >" prompt the following list of commands are available:

HELP	Lists the available commands.	
SHOW SERVER	Displays information about the server, including the number of sessions currently active or connected to it, the state of the server, and the CPU usage in terms of quantity used.	
SHOW SESSIONS	Displays a list of active sessions and their current states. The session states are:	
	CS_EVENT_WAIT	Waiting for an event. The event type is shown in parentheses: (LOCK), (DIO) - disk io, (BIO) - frontend communication.
	CS_MUTEX	Awaiting a semaphore (access to a system data structure).

	CS_COMPUTABLE	Runnable and waiting for a chance to run.
	CS_INTERRUPT	Interruptable, either a lock or frontend I/O request.
SET SERVER SHUT	Prevents additional server connections, and shuts the server down when the current sessions finish. This command can only be run by the ODT-DATA superuser.	
STOP SERVER	Stops the server immediately. Use this command only if absolutely necessary, for example, when a frontend program is hanging. This command can only be accessed by the ODT-DATA super user.	

The following commands use the session id to perform actions on a specific server session. The session id is displayed in the **iimonitor** utility with the **show sessions** command.

FORMAT *session id*	Gives a synopsis of the ODT-DATA information about a session.
REMOVE *session id*	Disconnects a particular session. This command can only be run by the ODT-DATA super user.
SUSPEND *session id*	Suspends a compute-bound session to allow technical support personnel to trace the problem.
QUIT	Terminates the IIMONITOR session.

Shared Memory and Semaphores Report Utility—csreport

Use the **csreport** utility to display shared memory and semaphore information for your installation. The **csreport** utility displays:

- the maximum number of servers configured for your installation
- shared memory and semaphore information

To invoke the **csreport** utility, type the following at the operating system prompt:

```
$ csreport
```

Output from the csreport Utility

The following is a example of output from **csreport**. The format is subject to change.

```
!Installation version 610008
!Max number of servers 16
!Description of shared memory for control system:
!key 0x49065A7A: size 8192 attach 00000000
!Description of shared memory for logging & locking system:
!key 0x49065A7C: size 327680 attach 0E000000
!Semaphore information for installation:
!sysV semid 0, num sems 21, used sems 19
! 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
!Event system: used space 2212, length space 8192
!Server info:
!server 0:
!inuse 0, pid 4917, connect id 0, id_number 0, semid 0
!shared memory:
!key 0xFFFFFFFF: size 0 attach 00000000
!server 1:
!inuse 1, pid 125, connect id 0, id_number 1, semid 0
!shared memory:
!key 0xFFFFFFFF: size 0 attach 00000000
!server 2:
!inuse 1, pid 128, connect id 0, id_number 2, semid 0
!shared memory:
```

```
!key 0xFFFFFFFF: size 0 attach 00000000
!server 3:
!inuse 0, pid 3770, connect id 2159, id_number 3, semid 12
!shared memory:
!key 0x49065A8E: size 122880 attach 00000000
!server 4:
!inuse 0, pid 360, connect id 0, id_number 4, semid 0
!shared memory:
!key 0xFFFFFFFF: size 0 attach 00000000
!server 5:
!inuse 0, pid 0, connect id 0, id_number 0, semid 0
!shared memory:
!key 0x00000000: size 0 attach 00000000
!server 6:
!inuse 0, pid 0, connect id 0, id_number 0, semid 0
!shared memory:
!key 0x00000000: size 0 attach 00000000
!server 7:
!inuse 0, pid 0, connect id 0, id_number 0, semid 0
!shared memory:
!key 0x00000000: size 0 attach 00000000
!server 8:
!inuse 0, pid 0, connect id 0, id_number 0, semid 0
!shared memory:
!key 0x00000000: size 0 attach 00000000
!server 9:
!inuse 0, pid 0, connect id 0, id_number 0, semid 0
!shared memory:
!key 0x00000000: size 0 attach 00000000
!server 10:
!inuse 0, pid 0, connect id 0, id_number 0, semid 0
!shared memory:
!key 0x00000000: size 0 attach 00000000
!server 11:
!inuse 0, pid 0, connect id 0, id_number 0, semid 0
!shared memory:
!key 0x00000000: size 0 attach 00000000
!server 12:
!inuse 0, pid 0, connect id 0, id_number 0, semid 0
!shared memory:
!key 0x00000000: size 0 attach 00000000
```

```
!server 13:
!inuse 0, pid 0, connect id 0, id_number 0, semid 0
!shared memory:
!key 0x00000000: size 0 attach 00000000
!server 14:
!inuse 0, pid 0, connect id 0, id_number 0, semid 0
!shared memory:
!key 0x00000000: size 0 attach 00000000
!server 15:
!inuse 0, pid 0, connect id 0, id_number 0, semid 0
!shared memory:
!key 0x00000000: size 0 attach 00000000
```

The Locking Facility Report—lockstat

Use the **lockstat** utility to display locking status information. The **lockstat** utility displays information about installation lock status.

To invoke the **lockstat** utility, at the operating system prompt, type:

```
$  lockstat
```

Output from the lockstat Utility

The following example is a typical output from **lockstat**.

```
==========Tue Mar 28 16:35:43 1989 Locking System Summary==========
        Create lock list          181          Release lock list          176
        Request lock             7520          Re-request lock            280
        Convert lock             9482          Release lock              7105
        Escalate                   11          Lock wait                    3
        Convert wait                0          Convert Deadlock             0
        Deadlock Search             3          Deadlock                     0
        Cancel                      0
------------------------------------------Locks by lock list--------------------------
Id: 00300011 Tran_id: 000000000000002E R_llb: 00000000 R_cnt: 0 Wait: 00000000  Locks:
(2,0/75) Status: NONPROTECT,NOINTERRUPT
        Id: 01B00009 Rsb: 00AD001D Gr: IX Req: IX State: GR PHYS(1) KEY(DATABASE,iidbdb)
        Id: 015F000D Rsb: 00350020 Gr: X Req: X State: GR PHYS(1)
KEY(PAGE,DB=00000001,TABLE=[1,0],PAGE=3)
Id: 0032000C Tran_id: 0000000000000030 R_llb: 00000000 R_cnt: 0 Wait: 00000000 Locks:
(0,0/75) Status: NONPROTECT,NOINTERRUPT
Id: 00330001 Tran_id: 0000000000000005 R_llb: 00000000 R_cnt: 0 Wait: 00000000 Locks:
(4,0/75) Status: NONPROTECT,NOINTERRUPT
        Id: 00390033 Rsb: 00380052 Gr: IX Req: IX State: GR PHYS(1)  KEY(SV_DATABASE,iidbdb)
        Id: 0155000D Rsb: 01970040 Gr: N Req: N State: GR PHYS(1)  KEY(DB_TBL_ID,iidbdb)
        Id: 01E80009 Rsb: 00910012 Gr: N Req: N State: GR PHYS(1)  KEY(OPEN_DB,iidbdb)
        Id: 01940009 Rsb: 01950009 Gr: IS Req: IS State: GR PHYS(1)
KEY(SV_PAGE,DB=00000001,TABLE=[1,0],PAGE=3)
Id: 00340003 Tran_id: 0000000000000004 R_llb: 00000000 R_cnt: 0 Wait: 00000000 Locks:
(0,0/75) Status: NONPROTECT
Id: 00350001 Tran_id: 0000000000000001 R_llb: 00000000 R_cnt: 0 Wait: 00000000 Locks:
(1,0/75) Status: NONPROTECT,MASTER,NOIN
        Id: 018C000C Rsb: 00910012 Gr: N Req: N State: GR PHYS(1)  KEY(OPEN_DB,iidbdb)
------------------------------------------Locks by resource--------------------------
Id: 00350020 Gr: X Conv: X Value: <0000000000000000> KEY(PAGE,DB=00000001,TABLE=[1,0],PAGE=3)
        Id: 015F000D Llb: 00300011 Gr: X Req: X State: GR PHYS(1)
Id: 00380052 Gr: IX Conv: IX Value: <0000000000000000> KEY(SV_DATABASE,iidbdb)
        Id: 00390033 Llb: 00330001 Gr: IX Req: IX State: GR PHYS(1)
Id: 00910012 Gr: N Conv: N Value: <0000000000000001> KEY(OPEN_DB,iidbdb)
        Id: 018C000C Llb: 00350001 Gr: N Req: N State: GR PHYS(1)
        Id: 01E80009 Llb: 00330001 Gr: N Req: N State: GR PHYS(1)
Id: 00AD001D Gr: IX Conv: IX Value: <0000000000000000> KEY(DATABASE,iidbdb)
        Id: 01B00009 Llb: 00300011 Gr: IX Req: IX State: GR PHYS(1)
Id: 01950009 Gr: IS Conv: IS Value: <0000000000000000>
Y(SV_PAGE,DB=00000001,TABLE=[1,0],PAGE=3)
        Id: 01940009 Llb: 00330001 Gr: IS Req: IS State: GR PHYS(1)
Id: 01970040 Gr: N Conv: IX Value: <0000000000000000> KEY(DB_TBL_ID,iidbdb)
        Id: 0155000D Llb: 00330001 Gr: N Req: N State: GR PHYS(1)
```

The Logging Facility Report—logstat

The **logstat** utility displays information about:

- the force-abort-limit set during installation
- the percent of the log file used
- open transactions
- open databases

To invoke the **logstat** utility, at the operating system prompt, type:

```
$  logstat
```

Output From the logstat Utility

The following example is a typical output from **logstat.**

Logstat utility information

```
==13-JUL-1988 15:04:13.71 Logging System Summary==============================
Database add                    387     Database removes        383
Transaction begins      5408            Transaction ends        5405
Log read i/o's                  506     Log write i/o's         7196
Log writes                      18174   Log forces              5840
Log waits                       8005    Log splits              197
Log group commit        5808            Log group count         5827
Check commit timer      5820            Timer write             5787
Inconsistent db                    0    Kbytes written 10551
----Current log file header-----------------------------------------
Block size:4096                 Block count:8000                Buffer count:1
CP interval:400                 Archive interval:7600           Abort interval:6400
Last Transaction Id:0000009114A00P0B
Begin: 8946:1668:248  CP: 8946: 2070:168                       End:<8946:2298:96>
Journal Window:,0,0..,0,0
Status: ONLINE,CPDONE

----List of active processes-----------------------------------------------------
ID       PID      TYPE    OPEN_   DB       WRITE    FORCE    WAIT   BEGIN        END
-------------------------------------------------------------------------------------
0010018  0000004F SLAVE   2       18121    5386     6380     5405   5404 65537
001001D  0000004C SLAVE   1                 0                0        803    2    1  65537
001001E  0000004B MASTER  1                 53               0        822    1    0  65537
```

```
----List of active databases----------------------------------------------------------
Id:FFFF0001      Database:($archiver, $ingres)     Status:NOTDB
        Tx_cnt:2 Begin:3  End:1    Read:0          Write:53 Force:0   Wait:1625
        Location:  None
        Journal Window:,0,0..,0,0<0,0,0>..<0,0,0>
        Id:002A0014      Database:(getto60,pixar)  Status:
        Tx_cnt: 1                 Begin:42 End:41   Read:0   Write:155    Force:57 Wait:63
        Location:        II_DATABASE:[INGRES.data.getto60]
        Journal Window:,0,0..,0,0<0,0,0>..<0,0,0>
        Id:01560015      Database:(company,pixar)   Status:
        Tx_cnt:0         Begin:3  End:3    Read:0           Write:3  Force:2   Wait:3
        Location:                 IIDATABASE:[INGRES.data.company]
        Journal Window:,0,0..,0,0<0,0,0>..<0,0,0>
----List of active transactions-------------------------------------------------------
Id:0F980016  Tran_id:0000009114A00495      Database:002A0014 Process:00010018
First:8496,4,96          Last:8496,11,2180         Cp:8495,7662,100
        Write: 11                 Split:0  Force:3  Wait:4
        Status: ACTIVE,PROTECT
Id:00010019  Tran_id:00000091149FEF81      Database:FFFF001 Process:0001001E
First:8495,7662, 1196     Last:8495,7662, 1456      Cp:8495,7260,836
        Write:53 Split:0  Force:0  Wait:822
        Status:INACTlVE
Id:0002001B  Tran_id:00000091149FEF82      Database:FFFF0001 Process:001001D
First:,0,0   Last:,0,0            Cp:8495,839,96
        Write:0  Split:0  Force:0  Wait:801
        Status:INACTIVE
============================= ============================================================
```

Determining Proximity to FORCE-ABORT-LIMIT

To determine how close you are to reaching the FORCE-ABORT-LIMIT, refer to previous example, "Logstat utility information," for a computation example and consider the following:

- The first boldfaced number highlights the "Abort interval" of 6,400. This figure refers to the number of blocks in the log file that must be filled before the FORCE-ABORT-LIMIT is reached.

- The second highlighted number is part of a numeric series following the word "Begin". The important value here (1,668) refers to the block marking the log file's Beginning of File (BOF).

- The third highlighted number is part of a numeric series following the letters "CP". The important figure (2,070) refers to the block marking the last consistency point. This consistency point contains a list of all open transactions and open databases between the BOF (1,668) and the CP (2,070).

- The fourth highlighted number is part of a numeric series following the word "End". The important figure (2,298) refers to the block marking the log file's End of File (EOF). This is the last block that contains transaction information.

To calculate the number of blocks in use, subtract the EOF from the BOF (2298 - 1668). Currently, 402 blocks are in use in the log file.

To determine how close you are to the FORCE-ABORT-LIMIT, subtract the total blocks used (402) from the abort interval (6400):

```
6400 - 402 = 5998
```

This log file still has plenty of room before reaching the FORCE-ABORT-LIMIT.

Also highlighted in this section is an area called "Status," which, in this case, states: "ONLINE, CPDONE." The status provided here is of the logging and recovery systems. **ONLINE** indicates that everything is fine. **CPDONE** indicates that a consistency point was taken and the status is fine. Other options that might appear as the status are:

cpneeded	The logging system is about to take a consistency point.
logfull	The log file is full and transactions will stall until the archiver process catches up.
force_ abort	The FORCE-ABORT-LIMIT has been reached; the oldest open transaction will be aborted.
recover	The recovery process is performing recovery.
archive	The archiver process is archiving journaled transactions to the journal files.
acp_ shutdown	The archiver is preparing to shutdown. (This status is displayed when you type "rcpconfig/shutdown".)

imm_ shutdown	The logging system has been told to shut down immediately. (This is displayed when you type "rcpconfig/imm_shutdown.")
start_ archiver	This is an important status indicating that the archiver has died and must be restarted by the DBA. The archiver will not restart automatically; if the DBA does not restart it, the log file will eventually reach full capacity.
purgedb	This status appears when a database has been closed by the last user who had it open; the archiver is archiving transactions that belong to this database.

Notifying Users of Imminent Shutdown

To shut down the system you must determine what databases are active and who the users are so that you can notify them of the imminent shutdown. In referring to the example of the logstat report displayed a few pages ago you can see the following.

The first database listed will always be owned by **$ingres**. In this case, **$archiver** is displayed as the database, but the status on this line indicates that **$archiver** is not a database. This entry shows that ODT-DATA is operating correctly.

The second database shown is "getto60", owned by "pixar". The ID for this database is 002A0014. Refer to the section entitled "List of active transactions". You will see that the first transaction listed belongs to "Database: 002A0014". The status of this database (getto60) is ACTIVE.

The third database shown is called "company,pixar", but since its ID number does not appear on the list of active transactions, "company" is not currently active. Both other transactions shown in this list are also inactive and belong to **$ingres**.

When EOF Is Less Than BOF

Because the log file is circular, it is not uncommon for the block marking the file's beginning to be a higher number than the block marking the file's end. The following example illustrates the "Current log file header" from the results of another **logstat** execution.

```
------Current log file header-----------------------------------------------------------
Block size: 4096                Block count: 8000                Buffer count: 1
CP interval:400                         Archive interval: 7600   Abort interval: 6400
Last Transaction Id:            0000009114A0049F
Begin: 8495:7662:100            Cp:8495:7662:100                 End:8496:16:96
Journal Window: ,0,0..,0,0
Status:                                 ONLINE, CP DONE
------List of active processes---------------------------------------------------------
```

- The first boldfaced number highlights the "Block count" of 8,000, the number of 4K blocks in the log file.
- The "Abort interval" here is 6,400, as it was in the previous example.
- The log file's Beginning of File (BOF) is 7,662.
- The consistency point (CP) is the same as the BOF (7,662). The amount of space occupied in the log file by open transactions since the last consistency point has not reached the five-percent threshold. A new consistency point is taken at the five percent threshold.
- The End of File (16) is smaller than the BOF (7,662). Such a discrepancy is possible because the log file is circular. To calculate the number of blocks in use, subtract the BOF from the block count (8,000), then add the EOF to the total, as in the following example:

  (8000 - 7662) + 16 = 354

  To determine how close you are to the FORCE-ABORT-LIMIT, subtract the total blocks used from the abort interval:

  ```
  6400 - 354 = 6046
  ```

Chapter 6
Installation Reference Material

This chapter provides reference information for installing ODT-DATA and for the server start up facility.

ODT-DATA Installation and Server Start Up Utility

A new server installation can be started by invoking **iistartup** along with one of the following flags from the operating system prompt:

-b $II_SYSTEM
To start an installation automatically each time the system is rebooted, include the command **iistartup -b $II_SYSTEM** in the UNIX system startup file. This is */etc/rc* or */etc/rc.local* at most sites. The server is started using the parameters saved in the file *rundbms.opt* if it exists; otherwise the default parameters will be used. This flag may not be used from the operating system prompt.

-i
The command **iistartup -i** can be used from the operating system prompt to start up a new server installation.The user is given the option to use the parameters saved in the *rundbms.opt* file, use default parameters, or to reconfigure *rundbms.opt*.

-n
The command **iistartup -n** can be used from the operating system prompt to start a new server or installation.The user is given the option to use the parameters saved in the *rundbms.opt* file or the default parameters, or reconfigure *rundbms.opt*.

iirundbms and DBMS Server Parameters

Access to a Release 6 database is controlled by the server (iidbms). **iistartup** invokes **iirundbms** to configure the server using parameters saved in the file *$II_SYSTEM/ingres/files/rundbms.opt*. If the server configuration option is chosen, the user is prompted with a selected list of **iirundbms** parameters that will be written to **rundbms.opt** and used for server startup. If parameters other than those prompted are desired,, edit **iirundbms** with the required parameter list.

DBMS Server Parameters

The following **iirundbms** parameters are recognized:

-connected_sessions *n*
The maximum number of server connections allowed. The default is 32, and *n* may not exceed 50.

-cursors_per_session *n*
The number of simultaneously open cursors per session. The default is 16.

-dmf.option *n*
The Data Manipulation Facility (DMF)manages the interface between the DBMS and stored data. The following **dmf** options are available:

cache_size
The number of individual buffers (single data pages) in the buffer manager. The default is (128 + 4 *sessions).

count_read_ahead
The number of group buffers (multiple data pages) in the buffer manager. The default is (4 + sessions/4).

size_read_ahead
The size of the group buffers used in READ_AHEAD operations. The default is 8.

tcb_hash
The size of TCB hash table used in lookups of table control blocks. Use values of power to 2 for optimal hashing operation. The default is 255.

wbstart Specifies the threshold for modified pages in the cache, after which write-behind starts. When there are wbstart-modified pages in the cache, the *write behind* threads wake-up and write-modified pages out of the cache until the number of modified pages drops below the **wbend** limit.The default is (**mlimit** - 15% of the cache size).

wbend Specifies the lower limit for modified pages. When it is reached the *write behind* threads go to sleep and wait for the **wbstart** limit to be reached. The default is(1/2 of the buffer manager (cache) size). If 1/2 of the cache is not less than **wbstart** then, the size defaults to (1/2**wbstart**).

flimit Specifies the minimum number of free pages that the buffer manager will try to keep available in the cache. When this limit is reached, the buffer manager begins doing synchronous writes of pages whenever needed. The default is (1/32 of the buffer manager (cache) size).

mlimit Specifies the maximum number of modified pages that can be left in the buffer manager. When this limit is reached, the buffer manager begins writing pages out of the cache each time a dirty page is unfixed. **Mlimit** must be greater than **wbstart**, if **wbstart** is specified and **mlimit** is not, then **mlimit** will default to halfway between **wbstart** and the size of the cache. The default is(3/4 of the buffer manager (cache) size).

memory Specifies maximum memory usage by the DMF for control blocks and caching. Calculate requirements based off total cache size (single and group buffers) and shade enough for control blocks. Keep in mind that each cache buffer requires 2K (ODT-DATA pages). The default is (500K, Value input * 1024).

-database _count *n*
The maximum number of open databases for the server. The default is the value of the **connected_session parameter.**

-dblist *dbname {dbname}*
Allows the specification of a list of databases that will be serviced by the DBMS server. By default all databases can be serviced by any DBMS server running in an installation. If you use the option **-dblist** *dbname* and the **-sole_server** option, that server will be the only one that can access the database *dbname*.

-fast _commit
Transaction I/O optimized by using a delayed write algorithm. Used only with **-sole_server** option.

-no public
Creates a private server. To access a private server, the environment variable II_DBMS_SERVER must be set to the value reported at start-up. The global value of II_DBMS_SERVER is not updated when this option is used.

-qef.option *n*
The Query Execution Facility (QEF) manages query plans and execution. The following **qef** option is available:

qep_size
Translates to the QEF data size to be used to calculate the maximum memory that QEF will use to build data segment headers, containing all the runtime information needed for the query to be executed. DSH_MAXMEM = 2048 + ((sessions *cursors) * qep_size)

-qsf *n*
The Query Storage Facility (QSF) manages shared memory between facilities. The following QSF option is available:

pool_size
Increases or resets the QSF memory pool by allocating *n* bytes of memory to it. The pool is used to store all the query objects, regardless of type. It serves as the facility that manages a session's memory use. The default size for the QSF pool is ((number of connected sessions * 40K)+ 60K).

-scf.option *n*
The System Control Facility (SCF) is the central controlling facility. The following SCF option is available:

row _estimate	This is the number of rows to expect on the first pass of a select or **retrieve**. This is used to build message blocks to send MDEs (message data elements) across GCAto frontends. This reduces the amount of formatting done before queries are executed. If insufficient buffers to send the data from the query are formatted, the additional buffers needed are built after the query completes.
-sole_server	Forces databases accessed by this server to refuse connection requests from other servers. This option should be used when possible to reduce the overhead in managing multiple server connections to common databases.
-stack_size *n*	The per-session stack size in bytes. The default is 32,768 bytes.
-write _behind *n*	Allows the creation of write behind threads in a DBMS server. *Write behind* threads write dirty pages out of the buffer manager when the cache starts getting full. This keeps users from having to do synchronous writes to free up space in the cache to read in a new page. They also wake up when consistency points are taken to help the *fast commit* thread flush all the dirty pages out of the cache. There are no rules to determine the optimal number of *write behind* threads to allocate and there is no way to dynamically add new *write behind* threads to the server once it is invoked. As a guideline, look at the load on the system and determine the number of writes per second that need to be done (figuring in the number of devices to be written to). Each *write behind* thread can perform approximately 20 I/Os per second up to the capacity of the operating system. The value for *n* must be some number greater than 0.

The following options control UNIX process parameters:

-maximum_working_set *n*	Sets the server's resident size (RLIMIT_RSS)in bytes.
-priority *priority*	Adjusts the server process priority (PRIO_PROCESS).

NOTE: These options may be abbreviated to the minimal unique prefix. For example, **-con** can replace **-connected_sessions.**

The DBMS Server Process Communications Address

The **iirundbms** command sets the *installation-wide* ODT-DATA environment variable II_DBMS_SERVER to the communications address of the server process. The communications address is an Internet socket number. Users can connect frontend programs to specific servers by setting the II_DBMS_SERVER, to the desired server's address, in their environment. Incorrectly setting the II_DBMS_SERVER environment variable will generate the following errors:

```
E_LC0001 GCA protocol service (GCA_REQUEST)
     failure with status E_GCfe05
     Connect failed

E_LQ0001 Failed to connect to DBMS session
```

The II_DBMS_SERVER Environment Variable

If you start additional servers with **iistartup** or **iirundbms** without specifying the **-nopublic** option, the value in II_DBMS_SERVER will be replaced with the new server's communications address. Then, connection to the old server will only be possible by explicitly setting II_DBMS_SERVER to the former value. To see the value of II_DBMS_SERVER for the installation, type:

```
$ ingprenv | grep II_DBMS_SERVER
```

ingprenv prints the names and values of all the ODT-DATA environment variables for an installation, including a single value for II_DBMS_SERVER.

WARNING: No record is kept of the previous value of II_DBMS_SERVER. Consult the ODT-DATA error log for old values of II_DBMS_SERVER.

DBMS Server Startup Troubleshooting

Check these items if the server fails to start:

- Are system shared memory and semaphore resources installed?
- Does the log file ($II_LOG_FILE/ingres/log/ingres_log) exist?
- Is the recovery process "dmfrcp" running?
- Is the archiver process "dmfacp" running?
- Is II_DBMS_SERVER set to the current server's communications address?
- Is the local node in the **/etc/hosts** file?

Logging and Locking Facility Parameters—rcpconfig

If the logging and locking configuration option of **iistartup** is chosen, **rcpconfig** *-init* is invoked to configure and initialize the log file. The user is prompted to enter logging and locking parameter values that will be written to **$II_SYSTEM/ingres/files/rcp.par** and used for configuration.The following sections contain descriptions of the **rcpconfig** parameters whose values must be set.

The rcpconfig Logging Parameters

ODT-DATA Release 6 builds a single circular log file per installation. The log file contains records used in aborted or incomplete transactions. It also contains records of completed transactions. These are taken by the archiver and placed in the corresponding journal files. The name of this file is *ingres_log*. It is located in the directory defined during installation by the environment variable II_LOG_FILE. This section contains information about the parameters used to configure logging.

- Number of log buffers in the memory (Default is 4.)

  This is the number of outstanding I/Os waiting to be put to the log file. These buffers are sized by the block size prompt,which follows. Sites with small transaction volumes may increase the number of log buffers and decrease the transfer block size,increasing throughput to the disk. Large transaction volume sites using larger block sizes and fewer log buffers log data faster.

- The maximum number of databases in logging system (Default is 32.)

  This is the maximum number of open databases that the logging system can handle at one time. The high side is the safe side for this value so that an unexpected database access attempt is not halted by a lack of available slots. This can also be used as a lockout method to prevent users from accessing additional databases. Gauge your answer accordingly.

- Maximum number of transactions in logging system. (Default is 32.)

  This is the maximum number of current (pending) transactions that can be handled by the logging system. Figure this value based on the amount of concurrent ODT-DATA processes on the system, servers(to include the recovery and archiver), and unique server-database connections. Gauge this value on the high side so that your system is able to start new transactions without making users wait until a slot becomes available.

- Block size of the log file. The legal block size is 4, 8, 16, or 32kbytes. (Default is 4)

  The log file is broken down into blocks which are used to transfer logging data (transaction information) from the log buffers in memory to the logging file on disk. This value is the block size of that unit of transfer per I/O. Sites processing large transaction volumes should use larger values, accomplishing the most throughput with fewer system actions.

- Log-full-limit in percentage. (Default is 95%.)

  Once the logging file reaches a certain percent of usage, the logger halts and backs out the oldest one and any that it finds at that same time stamp. Once this is accomplished, transaction processing begins again, on the assumption that available space in the file has been reclaimed. This is hard stopping point that prevents further transactions from being processed until sufficient logging files are cleared. This contrasts with the force abort limit which usually prevents this point from being reached. Large logging files should use the default value or higher since small values increase the likelihood large transactions will not proceed to completion.

- Percentage of log file for each consistency point. (Default is 5.)

  How often consistency points are written to the log file is determined by this value. The larger the logging file, the smaller the percentage should be, as this insures that there start time is not prohibitively long during the recovery process. This percentage can be set from 1 to 75.

- Enter the maximum consistent point interval for invoking archiver (Default is 19%.)

  The logging system uses consistency points to keep track of all active databases, transactions, and lock lists at certain intervals in the log file. This decreases the time required to restart the system after a crash, by finding the latest consistency point and resuming processing from there. This value tells the archiver that a certain number of consistency points have been written and it is time to wake up and begin archiving applicable data involved in that range. For example, if each consistency point involves five percent of the log file, a value of four here would wake the archiver each time twenty percent of the log file is available to be archived.

- Force-abort-limit in percentage. (Default is 80%)

  This soft failure point causes the oldest pending transactions to be aborted, preventing the log file from reaching the log full limit and causing a halt of all transaction processing until available log file space has been freed. Do not make this value too close to the log full limit value, or else the value of this parameter will be severely reduced because of the high probability that the log full limit will be reached anyway.

The rcpconfig Locking Parameters

This section contains information about parameters used to configure the shared memory locking.

- The size of the locks hash table (Default is 63.)

  In the ODT-DATA lock manager, there are two types of lock lookups. The first type uses the lock hash table to locate information about a lock owned by a specific user. This value creates the size of this lock lookup table used to determine the state of a given lock on a given resource. From this table the lock block is located and the associated resource block can be examined. Make this value greater than or equal to the resource hash table, because many types of locks can be queued on the same resource, but the same lock cannot refer to more than one resource.

- Size of the resources hash table (Default is 63).

  The second type of lock lookup is performed directly on the resource being locked. Through this table, information about specific resources is located and the associated locks on these resources can be determined. This value sets the size of this hashed lookup table.

- Maximum number of locks in locking system. (Default is 2000.)

  For transactions to process to completion and database access to be granted, there must be locks available in the ODT-DATA lock manager. This number should be a sum of all resources and locks required on the ODT-DATA system. Currently, ODT-DATA uses approximately 100-120 locks per user. Judge your answer according to your concurrency requirements. Sites that use very large transactions, with high max locks values, should set this value on the high side to insure that the ODT-DATA lock manager does not run out of locks. This will force a reconfiguring of the lock manager and its associated data structures.

- The maximum number of lock lists in locking system (Default is 128.)

  Lock lists are maintained on current transactions to speed processing and assure that MSTs are handled correctly. Locks accumulated by a pending transaction are chained together to help the transaction manager locate locking information required for completion of the task. There should be two lock lists per active user and five per server (to include the recovery and archiver processes).

- Maximum number of locks per transaction (Default is 150.)

  So that one transaction does not use all system locks or create an environment in which lock management overhead reduces performance, this parameter is required to place a cap on the number of locks a particular single statement or multi-statement transaction can own. Once this value is reached, escalation to table-level, locking takes place, reducing the number of locks taken by the transaction and letting the transaction proceed more quickly.

The rcpconfig Archiver and Recovery Shutdown Parameters

The **rcpconfig** command will shut down the recovery and archiver processes and deinstall shared memory when used with the following two options.

-shutdown	This will refuse all further connections and transaction processing, but allow those currently executing to finish and then execute a clean shutdown of the recovery and archiver processes. This is the friendly way to close down the recovery process and allow pending transactions to complete.
-imm _shutdown	This executes an immediate stop on all pending transactions and shuts down the archival and recovery processes. This should be used only when there is a critical need to close down the ODT-DATA recovery system that cannot be delayed until pending transactions finish.

The **rcpconfig** command with the **-shutdown** option is used by **shutserver** during installation shutdown.

ODT-DATA Installation Shutdown

The following describes **shutserver**, an automated shutdown utility and an emergency manual shutdown procedure.

The Installation Shutdown—shutserver

The shutdown utility can only be invoked by the superuser. From the operating system prompt type:

```
$ shutserver
```

You are then prompted through the steps to shut down your installation or sections of it.

Emergency Manual Installation Shutdown Procedure

The following describes an emergency procedure to shut down an ODT-DATA installation. Normally the server (**iidbms**), archiver (**dmfacp**), and recovery (**dmprcp**) processes should only be terminated with the **shutserver**, **iimonitor**, and **rcpconfig** utilities. If these utilities fail or hang, you may need to stop these processes using the UNIX **kill** command.

1. Log in as the "ingres" user.

2. Identify the ODT-DATA installation code of the installation you want to shut down by typing:

```
$ ingprenv | grep II_INSTALLATION
```

   The two-letter installation code will be displayed:

```
II_INSTALLATION=r6
```

3. Identify the installation's server(s) and their UNIX process id(s):

```
$ ps -e | grep iidbms
```

   The UNIX process id number and the installation code of the server(s) will be among the information displayed.

4. If a server has the UNIX process id "1912" and the correct installation code, shut it down using **kill** with the SIGQUIT signal:

```
$ kill  -QUIT  1912
```

   The following list of UNIX signals and expected effects is for information only. **SIGQUIT** is the preferred signal to use.

SIGHUP	Terminates the server if there are no active sessions.
SIGTERM	Terminates the server when all sessions finish.
SIGQUIT	Terminates the server immediately.

SIGKILL Terminates server process abnormally. Run **cscleanup.**

5. Once the server processes associated with the installation are stopped, the recovery and archiver processes can be killed. To kill the processes manually, identify their UNIX process ids and verify that they are the correct processes by their installation code:

```
$  ps -e | grep  dmfrcp
$  ps -e | grep  dmfacp
```

The UNIX process id number and the installation code of the recovery and archiver processes are among the information displayed.

6. If **dmfrcp** has process id "10469", **dmfacp** has process id "10471", and they have the correct installation code, then **kill** them using the SIGQUIT signal:

```
$  kill -QUIT  10469
$  kill -QUIT  10471
```

NOTE: If a server process was terminated abnormally with SIGKILL instead of SIGQUIT, **cscleanup** should be run. This program will attempt to release global system resources the server might have owned, such as shared memory and semaphores. The **csinstall** command should not be run again.

Once these steps are completed, the installation is completely shut down. Restart it using the command; **iistartup -n.**

WARNING
If the installation cannot be restarted from this point, you may have to shut down again and reinitialize the log file. This should not be done except as a last resort, as it will interfere with the recovery process.

Chapter 7
Troubleshooting with Log Files

This chapter explains how to use the ODT-DATA log files to troubleshoot ODT-DATA.

ODT-DATA Log Files

ODT-DATA creates log files where it writes information about your installation. The files described in this chapter are English text log files that you can use for troubleshooting.

NOTE: See Chapter 3 for information on the logging file associated with the logging and locking facility.

The Error Log

When you have an ODT-DATA problem, check the file *$II_SYSTEM/ingres/files/errlog.log*. Messages about your installation are appended to this log along with their dates and times. You find the following information in *errlog.log*:

- Archiver shutdown
- DBMS server startup and shutdown
- Error messages
- Warning messages

The *errlog.log* file is maintained by the system administrator. The *errlog.log* file will continue to grow until you shut down the installation and manually truncate the log.

The Archiver (dmfacp) Log

The file *$II_SYSTEM/ingres/files/II_ACP.LOG* is overwritten each time the archiver process is started up. ODT-DATA writes information about the current archiver process in this log, such as:

- Archiver startup
- Error messages
- Warning messages

The Recovery (dmfrcp) Log

The file *$II_SYSTEM/ingres/files/IIRCP.LOG* is overwritten each time the recovery process is started up. ODT-DATA writes information about the current recovery process in this log such as:

- Current logging and locking parameter values
- Error messages
- Recovery operations information
- Warning messages

The Installation (iibuild) Logs

During the ODT-DATA installation procedure, error messages and sometimes startup, and shutdown messages are saved in log files named for the processes whose output they store. The following log files are located in the directory *$II_SYSTEM/ingres/files*:

- **iigcn.log**
- **rcpconfig.log**
- **rundbms.log**

Appendix A
ODT-DATA Startup Files

When you invoke the **sql** command, the terminal monitor can read up to three different startup files. These can contain **sql** commands and macro definitions. Startup files can be installation-wide, database-specific or user-specific depending upon how they are invoked.

Installation-Wide Startup Files

The system startup file is automatically read when ODT-DATA is invoked with the terminal monitor. This file is included with your ODT-DATA installation and can be customized by the ODT-DATA system administrator to meet the specific requirements of your site.

The file *$II_SYSTEM/ingres/files/startsql* can be edited by the ODT-DATA system administrator to include SQL commands that are executed each time the terminal monitor is invoked.

For example, if the standard editor at your installation is not **/usr/bin/vi**, the **macro** {editor} can be redefined in the SQL **startup** file to invoke the proper editor. For more information about SQL terminal monitor macros, refer to the optional *ODT-DATA SQL Reference Manual.*

Database-Specific Startup File

Database-specific environment variables can be set to contain the full pathname of a system startup file. The file is read when the ODT-DATA terminal monitor is invoked for a particular database.

The environment variable DBNAME_SQL_INIT can be set to the full pathname of a file containing SQL commands. The filename is specified by the user. For DBNAME, substitute the name of the database for which the file is to be read. The database name must be specified in uppercase.

The **ODT-DATA** user can set these environment variables installation-wide by using the **ingsetenv** command as follows:

```
$  ingsetenv DBNAME_SQL_INIT path_name
```

The definitions for environment variables set with the **ingsetenv** command are stored in the file *$II_SYSTEM/files/symbol.tbl*. Type the command **ingprenv** to see the ODT-DATA environment variables and their values.

User-Specific Startup File

These definitions can also be set locally as user-specific environment variables. A good place to set them locally is in the user's ".login" or ".profile" file.

C shell example:

```
setenv DBNAME_SQL_INIT path_name
```

Bourne shell example:

```
DBNAME_SQL_INIT=path_name
export DBNAME_SQL_INIT
```

The environment variable II_SQL_INIT can be set to the full pathname of a file containing SQL commands. The file name is specified by the user.

A good place to set user-specific environment variables is in the user's .login or .profile file.

C shell example:

```
setenv II_SQL_INIT path_name
```

Bourne shell example:

```
II_SQL_INIT=path_name
export II_SQL_INIT
```

Appendix B

Authorizing User Access to ODT-DATA and Databases

ODT-DATA maintains a special "database" named **iidbdb**. It is used to store information about databases and authorized ODT-DATA users, and specifies the databases that can be accessed by specific users. The information in the **iidbdb** is updated by ODT-DATA when a database is created or destroyed. The **iidbdb** is consulted to validate a user's request to use ODT-DATA, to validate a user's request to use a particular database, and to determine where a particular database is stored. The **iidbdb** is itself an ODT-DATA database owned by the ODT-DATA system administrator.

The information in the **iidbdb** regarding the databases and where they are located is automatically maintained by ODT-DATA. This information is updated by the **createdb** and **destroydb** commands. This information is further affected by commands that relocate tables in a database into different directories or filesystems.

The information in the **iidbdb** about authorized ODT-DATA users and the databases they can access is maintained by the ODT-DATA system administrator with the **accessdb** program, described in this chapter.

ODT-DATA users other than the ODT-DATA system administrator can use the **catalogdb** command to view this information. This command is described in the optional *ODT-DATA SQL Reference Manual.*

Database Access

For user Bob to access database "xyz," at least one of the following conditions must be true:

- Bob is the data base administrator (DBA) for "xyz."
- "xyz" is globally accessible.

- Bob has been explicitly authorized by the xyz DBA to use database "xyz."
- Bob has the ODT-DATA superuser flag set in his user entry, and uses the **-u** (user) flag on the command line

By default, databases are globally accessible when they are created. The **-p** (private) flag of the **createdb** command must be used to create a private database. Operations such as changing the global access status of a database, extending a database to different locations, or authorizing a particular user to access a specific private database, may be performed by the ODT-DATA system administrator using the **accessdb** command. User authorization for database access is accomplished by adding the user to the table of users authorized to access the database, or by adding the database to the list of databases the user is authorized to access.

Database access does not automatically authorize access to tables in the database. Only the DBA for that database may authorize user access to shared tables with the SQL **create permit** command.

Defining the Terminal

If you are running in ODT-DATA/WINDOWVIEW the *termcap* variable is set.

The **accessdb** command must be run on a cursor-addressable terminal whose description is contained in the *$II_SYSTEM/ingres/files/termcap* file. The environment variable TERM (or TERM_INGRES) sets the terminal definition for use by the forms products. For example, if you want to abandon ODT-DATA/WINDOWVIEW system and use your terminal with function keys active in the C shell environment, you can identify your terminal, in your *.login* or *.cshrc* file, to **accessdb** by typing the following command:

```
setenv TERM_INGRES ansi
```

To identify your terminal in the Bourne shell environment, in your profile file, type the following commands:

```
TERM_INGRES=ansi
export TERM_INGRES
```

You can specify any of the valid terminal codes listed in the optional manual *Using ODT-DATA Through Forms and Menus.*

Invoking accessdb

After you define your terminal, start up **accessdb** by typing the following command at the operating system level:

```
$     accessdb
```

Remember that only the ODT-DATA System Administrator, "root" and other accounts you set up in the **iidbdb** with ODT-DATA superuser privilege are allowed to use **accessdb**.

Using accessdb

Because **accessdb** is a forms-based program, it is important to understand how to enter data into forms and select menu items before using it. If you are unfamiliar with the ODT-DATA forms products, start out using QBF™ or VIFRED™ to become familiar with forms applications. Refer to the book *Using ODT-DATA* in the User's Guide for more information.

When **accessdb** starts up, it displays a main menu of commands. The process of using **accessdb** consists of selecting a command from this main menu, moving on to another form or menu, optionally selecting another command, and exiting by selecting the **Quit** command. The execution of commands causes **accessdb** to display a new form with menu items appropriate to the function chosen. To return to the main menu from any form, enter the **End** menu item. Other menu items may also return you to the main menu after performing their sub-function.

All menus within **accessdb** contain an entry for **Help**. If you are in doubt about the meaning of a form that you have selected, select the **Help** menu item to get a quick reminder.

Functions in accessdb

The main menu in **accessdb** contains the following commands:

Command	Function
Catalog	Submenu of additional operations.
Database	Summarize information about a database
ExtendDB	Extend a database to a new volume.
LocationName	Create/Examine a *locationname*-area mapping.
User	Create/Modify/Delete an ODT-DATA user or specify the databases a user may access.
Help	Print help about the top level menu.
Quit	Exit from the **accessdb** program.

Database Function

The **Database** command is used to view and update information about a single database. When the **Database** command is selected from the main menu, you are prompted to enter an existing database. A form describing the database is displayed or an error message is generated if the database does not exist. To return to the main menu, select the **End** menu item. Help is available by selecting the **Help** menu item. The information displayed on the form includes the fields listed in the following table.

NOTE: Only the user table and the global access flag may be updated using the **Database** command.

Field Name	Mode	Description
Database name	read	Name of database.
Owner	read	Owner name.
Database location	read	*Locationname* upon which the ODT-DATA database resides (from **createdb**).

Field Name	Mode	Description
Checkpoint location	read	*Locationname* upon which ODT-DATA checkpoints reside (from **createdb**).
Journal location	read	*Locationname* upon which ODT-DATA journal files reside (from **createdb**).
Global access	read/write	Set "no" for private databases, and "yes" for globally accessible databases.
Database users	read/write	For private databases, a list of users explicitly authorized to access database. This is a table field, which can be edited. To delete a user, move to the row and press **Return**. To add another user, move to an empty row and enter the new name.

The **Database** form allows you to change both the global access status of a database and its list of explicitly authorized users. To do so, once the database form is on the screen, edit the form to reflect the changes, then select the **Save** menu item. To void the changes you made, select the **End** menu item instead.

ExtendDB Function

The ExtendDB function is used to extend a database to other locations. To locate tables outside the default area or the area corresponding to the *locationname* specified on the **createdb** command line, the database must be extended to a previously defined location.

ODT-DATA Locationnames

Locationnames are labels that are mapped to a UNIX directory. Each *locationname* maps to a single area. The area is described with the full path name of a directory. *Locationnames* are chosen by their creator and must follow the ODT-DATA naming convention. They must be alphanumeric and begin with a letter. The maximum length of a *locationname* is 32 characters. The area designation can be up to 240 characters including the "/" character and must follow the UNIX syntax for directory names.

Locationnames are used by the **createdb** and **finddbs** utilities. If a *locationname* is not specified the default *locationname* is used.

Each ODT-DATA installation has a set of default *locationnames*. These defaults are ii_database, ii_journal and ii_checkpoint. The *locationnames* map to the ODT-DATA environment variables II_DATABASE, II_JOURNAL, and II_CHECKPOINT, respectively. These defaults are mapped to the areas these environment variables are set to during the installation procedure. They *cannot* be changed.

To Extend a Database

Select the **ExtendDB** menu item from the main menu. Enter the login of the DBA of the database to be extended. A form then displays two tables: one listing existing *locationnames* and database names, and one in which you can add new database extensions. The names currently in the first table are those databases that have been extended to the specified *locationname*. One database may be extended to many locations.

To extend a database, both the database and *locationname* must exist. The database must be owned by the specified DBA, and the corresponding *locationname* must be available for databases as defined in the **LocationName** menu item in the main menu. To extend a database to a location, move the cursor to the table of new database locations, and enter the database name and the corresponding *locationname* in the proper columns. When you are done extending databases for a particular DBA, enter the **Save** menu item. The data in the table is then checked. If any of the data in the table are invalid, or if a database extension or restriction is not allowed, the cursor is positioned on the row containing the offending data. You can then correct the data and reenter the **Save** menu item. When the changes are accepted, you are returned to the main menu.

LocationName Function

The **LocationName** function is used to view or add the *locationnames*. Each execution of the **LocationName** function operates on a single *locationname*.

When the **LocationName** function is selected from the main menu, the system prompts you for a *locationname*. Enter a new (i.e., undefined on the system) or existing name. If you enter an existing name to view, a form appears displaying the current attributes of the *locationname*. If the *locationname* being entered is new, you are prompted to create a new location. If the answer is "yes" a blank form appears with the default values filled in. Fill in the appropriate data fields on the form.

Adding a New Locationname

The new *locationname* appears on the top of the form. A table-field displays the databases currently using that location. (This should be empty now.) Fill in the "area" field with the full pathname for the new area *locationname*. Before database, journal, or checkpoint usage permission can be granted for a **locationname**, the corresponding ODT-DATA directory and subdirectories must exist and be accessible by ODT-DATA. Fill in the usage permissions. The privileges are as follows:

- **Data Bases** allows database relations to reside on the specified *locationname*. The default is "yes."
- **Journals** allows journals to reside on the specified *locationname*. The default is "no."
- **Check Pts** allows checkpoints to reside on the specified *locationname*. The default is "no."

Select or create a directory to become the new ODT-DATA area. The directory must have the following permissions:

- read, write, and execute for "owner"
- read and execute for "group"
- read and execute for "world"

Create a subdirectory named "ingres." This directory must be owned by "ingres" and have the following permissions:

- read, write, and execute for "owner"
- read and execute for "group"
- read and execute for "world"

Create a subdirectory named "data," "jnl" or "ckp." This directory must be owned by "ingres" and have the following permissions:

- read, write, and execute for "owner"
- no permission for "group"
- no permission for "world"

Create a subdirectory named "default." This directory must be owned by "ingres" and have the following permissions:

- read, write, and execute for "owner"
- read, write, and execute for "group"
- read, write, and execute for "world"

To create the subdirectories needed for data in the area /install/new_area, follow this procedure:

```
$  mkdir  /install/new_area
$  mkdir  /install/new_area/ingres
$  mkdir  /install/new_area/ingres/data
$  mkdir  /install/new_area/ingres/data/default

$  chown  ingres  /install/new_area
$  chown  ingres  /install/new_area/ingres
$  chown  ingres  /install/new_area/ingres/data
$  chown  ingres  /install/new_area/ingres/data/default

$  chmod  755  /install/new_area
$  chmod  755  /install/new_area/ingres
$  chmod  700  /install/new_area/ingres/data
$  chmod  777  /install/new_area/ingres/data/default
```

To create directories and subdirectories for journals or checkpoints, follow the same procedure substituting "jnl" or "ckp" for "data" in the procedure.

When the form accurately describes the new *locationname*, select the **Save** menu item. If you decide not to create the locationname, select the **End** menu item.

Modifying a Locationname

When you exit the *locationname* form, only the usage permissions can be modified. The mapping between the *locationname* and area is permanent.

User Function

The **User** function can add, modify, or delete access to ODT-DATA by a user and can identify the databases that a user may access. The **User** function can be used to display an existing user's authorization information. Each invocation of the **User** function operates on a single user.

When the **User** function is selected from the main menu, you are prompted for a user name. Enter either the existing ODT-DATA user or the login of a user to be added. A form describing the user is displayed.

Adding a New User

When you enter a new user, you are prompted to verify your intent to create a new user. Enter yes and the form with the new user name, with default permissions is displayed.

Fill in the user information on the form that appears. The privileges are as follows:

- **Create database** permission allows a user to create new databases. The default is "yes."
- **Update system catalog** permission allows a user to directly update system catalogs (attribute, relation, etc.) using SQL. This is rarely needed. The default is "no."
- **Set trace flag** permission allows a user to set the debugging trace flags within ODT-DATA. The default is "no."
- **Superuser** permission allows a user to impersonate any other user in the system or to run the **accessdb** program.

Change the default privileges provided by entering a "y" or "n" next to the privilege shown. The first database owned by the user, is a read-only table field. The other table field, of the databases that the user can access, is filled in with the names of private databases you wish the user to access. Enter a database name, press **Return**, enter another database, and so on. When finished, press the **Menu** key to bring up the menu, and select the **Save** menu item to create the new user. If you decide not to create the user after filling in part of the form, use the **End** menu item.

Modifying an Existing User

To examine or modify an existing user's permissions, select the **User** command from the main menu and enter the user's login when prompted. Edit the form, including changes to the databases the user is explicitly authorized to access in the "May Access" table field. The "Owns" table field is read-only. Only the **createdb** and **destroydb** commands can be used to add and delete databases. Select the **Save** menu item to save the changes. To examine the user's permissions, or decide not to save your changes, specify the **End** menu item instead.

Deleting a User

To delete the authorization entry for an ODT-DATA user, select the **User** command from the main menu and enter the user's login. When the user's information form appears on the screen, issue the **Delete** menu item. If you decide not to delete the user, specify the **End** menu item instead.

Deleting a user is not permitted if that user owns any databases.

Catalog Function

In addition to establishing database extensions, locationname-area mappings and ODT-DATA users, the **accessdb** utility enables you to view current database access entries. The **Catalog** operation in the main **accessdb** menu calls a submenu of functions that display the data in separate frames.

The **Catalog** submenu contains the following commands:

Command	Function
Databases	Display a read-only table of databases.
LocationNames	Display the read-only table of *locationname*-area mappings.
Users	Display a read-only table of users.
Help	Get help about the operations in this menu.
End	Return to the **accessdb** main menu.

Databases Catalog Function

The **Databases** command displays a read-only table field containing a summary of each ODT-DATA database. For each database, the database name, owner, and global access flag are displayed. To change information about a database, return to the **accessdb** main menu and select the **Database** command. To scroll up and down in the table field, use the techniques described in the book *Using ODT-DATA* in the User's Guide.

To return to the **Catalog** menu, select the **End** menu item. To return to the main menu, select **End** again.

LocationNames Catalog Function

The **LocationNames** function displays a read-only table field containing each *locationname*-area mapping. Scroll through the table field to view its entire contents.

To return to the main menu, select the **End** menu item.

Users Catalog Function

The **Users** function displays a read-only table field containing a summary of each ODT-DATA user. For each user, the login and user access privileges are displayed. The display is a table that may be scrolled up and down to view its full contents. You cannot modify this information. To *change* information about a user, return to the **accessdb** main menu and select the **User** function.

To return to the main menu, select the **End** menu item.

Summary of Accessdb

The **accessdb** command can *only* be used by the ODT-DATA System Administrator or ODT-DATA superusers to modify, add, delete or list ODT-DATA users and database access privileges. It uses a forms-based interface, so **accessdb** must be run on a supported video terminal. Other ODT-DATA users cannot use **accessdb**, but they may use the **catalogdb** command, described in Chapter 4 of the optional *ODT-DATA SQL Reference Manual*. The **catalogdb** command displays *in read-only mode* information similar to **accessdb**.

Appendix C
ODT-DATA Environment Variables

Environment variables require definitions for your ODT-DATA installation. Certain installation-wide parameters should only be used by the "ingres" user in symbol tables. Other variables can be defined or redefined by individual users to customize their local ODT-DATA environment.

Setting Installation Wide Environment Variables

Environment variables are made system-wide by the "ingres" user, who can use the **ingsetenv** command to write them to the ODT-DATA symbol table.

For example, to set the environment variable ING_EDIT for an installation, you would log in as the "ingres" user and type:

```
$ ingsetenv ING_EDIT /usr/bin/vi
```

You can display all installation-wide variables by typing:

```
$ ingprenv
```

Setting User Defined Environment Variables

ODT-DATA environment variables can be set or reset by users in their local environment using UNIX commands. For example, one variable usually set in the user's environment is TERM_INGRES. It defines the ODT-DATA termcap definition to be used by the Forms System. To reset it in your local environment:

C shell:

```
% setenv TERM_INGRES vt100f
```

Bourne shell:

```
$ TERM_INGRES=vt100f
$ export TERM_INGRES
```

Users can display the values set in their environment:

```
$ env
```

Environment variables set in a user's local environment supersede the variables set in the symbol table. Set user-defined variables in the user's **.login** or **.profile** file.

Environment Variable List

Environment variables include the following:

- *DBNAME*_ING can be set installation wide or locally, to the full pathname of a file containing SQL commands. The commands are processed when a user connects to *DBNAME*, using the ODT-DATA SQL terminal monitor. Setting this variable is the equivalent of a user executing "\i file" in the terminal monitor each time they connect to *DBNAME*. Note that *DBNAME* is the name of the database and must be specified in uppercase.

- *DBNAME*_SQL_INIT may be set installation-wide or locally, to the full pathname of a file containing SQL commands. The commands are processed when a user connects to *DBNAME*, using the ODT-DATA SQL terminal monitor. Setting this variable is the equivalent of a user executing "\i file" in the terminal monitor each time they connect to *DBNAME*. Note that *DBNAME* is the name of the database and must be specified in uppercase.

- II_AUTHORIZATION is set during installation. The variable is set by the ODT-DATA System Administrator, during the **iibuild** procedure. It can only be updated by the "ingres" user using the **ingsetauth** command.

- II_CHECKPOINT is set to the full pathname for the default checkpoint location, ii_checkpoint. This variable is set by the ODT-DATA system administrator during the **iibuild** procedure, and it may not be changed, even during installation updates. Specific databases may designate alternate locations for checkpoints as a parameter to the **createdb** command.

- II_CONFIG is set to the full pathname of the ODT-DATA files directory during the ODT-DATA installation procedure.

- II_DATABASE is set to the full pathname for the default database location, ii_database. This variable is set by the ODT-DATA system administrator during the **iibuild** procedure and it may not be changed, even during installation updates. Specific databases may designate alternate locations as a parameter to the **createdb** command.

- II_DATE_FORMAT defines the format style for date *output*. Default is the **US** value with an output format of *dd-mmm-yyyy*. Default legal *input* formats for dates are

  dd-mmm-yyyy
  mm/dd/yy
  mmddyy

If the environment variable is set, it replaces one of these formats with an alternative format. The following are valid settings for II_DATE_FORMAT:

Value	Alternative Input Format	Replaces Input
US	*dd-mmm-yyyy*	
ISO	*yymmdd*	*mmddyy*
SWEDEN or **FINLAND**	*yyyy-mm-dd*	*dd-mmm-yyyy*
MULTINATIONAL	*dd/mm/yy*	*mm/dd/yy*
GERMAN	*dmmyy*	
	ddmmyy	
	dmmyyyy	
	ddmmyyyy	

In all cases, the alternative input format becomes the default output format. The three **US** default input formats listed above are unaffected by alternative settings and remain valid date input formats.

- II_DBMS_SERVER points to the DBMS server to which the user's process will connect to. The default value for any server connection request points to the primary DBMS server, which is created at startup.
- II_DECIMAL specifies the one character used to separate fractional and nonfractional parts of a number. Default value is the period (.), as in 12.34. Alternatively, the comma (,) can be used, as in 12,34. Only (.) and (,) are allowed.
- II_DML_DEF defines the default query language for an installation as **SQL.** RBF uses this variable.
- II_ERSEND is set during the ODT-DATA installation procedure to the full pathname of the *errlog.log* file.

- II_FILES is set to the full pathname of the ODT-DATA files directory during the ODT-DATA installation procedure.

- II_GCN*xx*_PORT contains the connect address of an installation's name server, where *xx* is its two-letter installation code (II_INSTALLATION).

- II_HELP_EDIT, if set to any value, adds an extra operation, **Edit**, to menus encountered in help text screens in the ODT-DATA forms systems. The **Edit** operation enables users to edit the help text in the screen with the text editor defined by ING_EDIT.

- II_INSTALLATION is a two-character code used to define a particular ODT-DATA installation. It should be defined at the installation level, using the **ingsetenv** command. It should never be defined at the user level.

- II_JOURNAL is set to the full pathname for the default journal location ii_journal. This variable is set by the ODT-DATA system administrator during the **iibuild** procedure and may not be changed, even during installation updates. Specific databases may designate alternate locations for journals as a parameter to the **createdb** command.

- II_LG_MEMSIZE is set to a value that will be used for the size, in bytes, of the locking and logging shared memory segment, created when **csinstall** is called by **iistartup**. The size of the segment is fixed, until the next time **csinstall** is executed. II_LG_MEMSIZE is calculated from the locking and logging parameters you selected for your ODT-DATA installation, when **iistartup** calls **iirun** to start the recovery process. The value of II_LG_ MEMSIZE should never be set below the default of approximately 200K.

  In the current release of the system, the size of the LG/LK shared memory segment defaults to approximately 200K. This segment's size is fixed once it is created by **csinstall**. The size may increase the next time **csinstall** is called. When the **dmfrcp** process starts up, it reads the locking and logging parameters that the user has specified during the **iibuild**, and calculates the maximum amount of memory that may be needed to support those parameters. The RCP sets a system-level environment variable II_LG_MEMSIZE to this value (represented in number of bytes). Until shutdown the rcp will continue to use the default size segment, possibly returning "out of (locking/logging) memory" error conditions on some queries. Subsequently, whenever the system is restarted (using **iistartup**, and thus calling **csinstall** to recreate the locking/logging segment), the LG/LK segment will be created with whatever size the variable II_LG_MEMSIZE is set to. II_LG_MEMSIZE should never be set below the default of approximately 200K.

- II_LOG_FILE is set to the parent directory of *ingres/log/ingres_log*. It determines the location of the installation-wide logging file. The **iibuild** procedure prompts you for this.

- II_MSG_TEST is set to false during the ODT-DATA installation procedure, and should not be changed.

- II_MONEY_FORMAT defines the format of money output. You can change output by setting the variable to a string with two symbols separated by a colon (:). The symbol to the left of the colon must be an "L" for a leading currency symbol, or a "T" for a trailing symbol. To the right of the colon, put the currency symbol you want prepended or appended to the amount. Examples follow:

Environment Variable Definition	Result
L:$	$100
T:DM	100DM
T: FF	100 FF

- II_MONEY_PREC shows the number of digits of precision to be used in the default representation of money data. The default is two digits (for decimal currency). Valid choices are 0, 1 and 2.

- II_NUM_SLAVES specifies the number of slave processes that a DBMS server will create to do disk operations. The default is two, but systems with many or faster disk drives have to increase the number of slave processes. For example:

```
$ ingsetenv II_NUM_SLAVES 4
```

  The above example allows each new server that is started to have four slaves. Stop and restart servers to reset the number of slaves. The maximum supported value is 10, and the minimum is 0. Setting this variable to 0 will degrade performance substantially.

- II_PATTERN_MATCH determines the set of pattern-matching characters for the QBF qualification function.

- II_PRINTSCREEN_FILE is provided for use with the **printscreen** function. It specifies a default file name for the output file of the **printscreen** function. If this variable is set and no file name is sent to the **printscreen** function, then no prompt is given and this file name is used. If not set, the user is prompted for a file name. If the file name *printer* is specified for II_PRINTSCREEN_FILE, the screen depiction is sent directly to the line printer. For more information, refer to the optional manual *Using ODT-DATA Through Forms and Menus.*

- II_SORT specifies the area that the location **ii_database** is mapped to. This is the default location, in which ODT-DATA creates temporary sort files. Sort files are created during the processing of ODT-DATA/SQL commands like **modify** and **create index**.

- II_SQL_INIT may be set locally, to the full pathname of a file containing SQL commands. When a user with this variable set connects to the ODT-DATA SQL terminal monitor, the commands in the named file are processed. Setting this variable is the equivalent of executing *\i file* in the terminal monitor each time a user connects to a database.

- II_SYSTEM specifies the parent directory of *ingres*. This environment variable should not be changed unless ODT-DATA is reinstalled.

- II_TEMPLATE is set during the ODT-DATA installation procedure to the full pathname of the database *template* directory.

- II_TERMCAP_FILE specifies an alternative termcap file to use. This file must be in termcap file format.

- II_THOUSANDS is set to a one-character symbol indicating the separator between thousands in numbers. Choices are the comma (,) and the period (.) The default is a comma.

- II_TMPDIR is used by ODT-DATA to locate temporary sort files. These are created during the processing of many ODT-DATA commands: queries (such as retrieval with joins and sort by clauses), the modify command and index, to name a few. II_TMPDIR is not for temporary files created by the ODT-DATA frontend and programs. Use II_TEMPORARY to locate these files elsewhere.

- ING_EDIT specifies the default editor spawned by various editor commands. The default for the entire installation is set during the ODT-DATA installation procedure. Users can also set this in their local environment.

- ING_PRINT specifies the default printer command issued by the **Print** function in ODT-DATA. The default is PRINT.

- ING_SET may be set installation-wide or locally, to a quoted string. The string must be 64 characters or less, or it will be invalid. It may contain either **set** commands separated by colons, or the word "include" followed by the full pathname for a file of **set** commands. If the variable is defined, the **set** commands are executed when any ODT-DATA frontend, including the terminal monitor, connects to a DBMS server. For example:

  C shell

  ```
  % setenv ING_SET "set autocommit on;
        set result_structure 'cbtree'"
  ```

  Bourne shell

  ```
  $ ING_SET="set autocommit on;
        set result_structure 'cbtree'"
  $ export ING_SET
  ```

- ING_SET_*DBNAME* may be set installation wide or locally, to a quoted string. The string must be 64 characters or less, or it will be invalid. It may contain either **set** commands separated by colons, or the word "include" followed by the full pathname for a file of **set** commands. If the variable is defined, the **set** commands are executed when any ODT-DATA frontend including the terminal monitor, connects to a DBMS server for *DBNAME*. Note that *DBNAME* is the name of the database and must be specified in uppercase.

- ING_SHELL, if defined, contains the pathname of the shell that ODT-DATA/MENU uses when the **shell** operation is invoked.

- ING_SYSTEM_SET may be set installation-wide, to a quoted string. The string must be 64 characters or less, or it will be invalid. It may contain either **set** commands separated by colons, or the word "include" followed by the full pathname for a file of **set** commands. If the variable is defined, the **set** commands are executed when any ODT-DATA frontend, including the terminal monitor connects to a DBMS server.

- INIT_INGRES may be set locally, to the full pathname of a file containing SQL commands. When a user with this variable set connects to the ODT-DATA SQL terminal monitor, the commands in the named file are processed. Setting this variable is the equivalent of executing "\i file" in the terminal monitor each time a user connects to a database.

- TERM is the terminal description for the terminal upon which you are executing one of the ODT-DATA forms-based products, such as QBF or VIFRED. See *Using ODT-DATA Through Forms and Menus* for a complete list of supported values. This environment variable is defined in the user's **.login** or **.profile** file.

- TERM_INGRES contains the terminal designation for the terminal upon which you are executing one of the ODT-DATA forms-based products, such as QBF or VIFRED. See the optional manual *Using ODT-DATA Through Forms and Menus* for a full list of supported values. This environment variable is most conveniently set in a **.login** or **.profile** file. TERM_INGRES takes precedence over TERM in defining terminal type to ODT-DATA and allows TERM to be defined differently for use by other UNIX programs such as *vi*.

Appendix D
ODT-DATA System Recovery

The recovery tools, **rolldb** and **finddbs**, are provided by ODT-DATA to recover from a database or installation corruption. This section briefly discusses when to use each of these tools, and describes the **finddbs** command. See the optional *ODT-DATA SQL Reference Manual* for additional information about these commands.

The **rolldb** command can regenerate a database from a static backup called a checkpoint. If your installation keeps a dynamic record of changes to the database called *journals*, they can be used to restore the database up to the time of the system failure.

If system failure causes corruption of the information in the *iidbdb*, the **finddbs** command can be used to recover the locations of databases. This information is stored in the *iidbdb* system catalog, *iidatabase*. The **finddbs** command allows users to search any directories they choose for databases and enter them into the *iidatabase* table.

Using finddbs

The syntax of the **finddbs** command is as follows:

finddbs [**-a** | **-r**] [**-p**]

The arguments are the following:

-a	This runs **finddbs** in "analyze" mode, but simply writes intended updates to the terminal. Use this option if you suspect *iidbdb* database catalog is out of order.
-r	This runs **finddbs** in "replace" mode, actually inserting databases located by the program in the *iidbdb*.
-p	makes all located databases private, except for the *iidbdb*.

The **finddbs** command helps recover ODT-DATA when *iidbdb* has been destroyed. Only the ODT-DATA system administrator can use it. The **finddbs** command runs in analyze mode, which is the default, or replace mode. Analyze mode is selected with the **-a** flag, while replace mode is selected with the **-r** flag. Analyze mode is a read only mode that informs you about possible errors in the *iidbdb iidatabase* table with a new table formed by scanning the ODT-DATA directories on a set of devices for databases. The use of replace mode is not recommended unless the *iidbdb* **iidatabase** table is in error.

When **finddbs** starts, it first builds a list of found databases. The list is formed by scanning the $II_SYSTEM/*ingres/data/default* directory. You are then prompted for any other directories to search. If you respond with a directory name not preceded by a slash, **finddbs** looks in $II_SYSTEM/*ingres/data/name*, where *name* is the name you specify. If you specify a directory name beginning with a slash, **finddbs** looks in */name/ingres/data/default*. Enter an empty line to exit this scan phase. It is important to search all directories that contain databases. If your installation creates all its databases in the default directory, then the default search will suffice. In non-default directory names are specified on the **creatdb** command line at your installation, or you have extended databases to alternate locations with **accessdb**, then you must specify all such directory names so that **finddbs** can find the databases they contain.

NOTE: Use **finddbs** in a peer ODT-DATA installation that is not located in the home directory of the ODT-DATA system administrator. Define the ODT-DATA system administrator's environment to run the peer installation and use **finddbs** as described.

Use the analyze mode of **finddbs** if you suspect that the *iidatabase* table of the *iidbdb* is in error. The output of analyze mode consists of two lists. The first gives names of databases present on disk but not contained in the *iidatabase* table. If analyze mode reports differences between the found database list (built during the scan phase) and the *iidbdb iidatabase* table, you may decide to run **finddbs** in replace mode.

The **finddbs** command replace mode replaces the contents of the *iidbdb iidatabase* table with a new table consisting of the databases found during the directory scan phase. Before running **finddbs** in replace mode you should first run it in analyze mode to see what changes would be performed by replace mode. By default, replace mode sets the access status of all databases to "global". The **-p** flag causes all databases to be made private, except for the *iidbdb*.

Examples of the **finddbs** command are:

```
#     Run finddbs in analyze mode to examine
#     directories and iidatabase table in the iidbdb.

$     finddbs -a

#     Replace the contents of the iidatabase system
#     catalog with databases found running this
#     command.

$     finddbs -r
```

In both cases, **finddbs** prompts you for the directories to check for ODT-DATA databases.

WARNING

When the **finddbs** command is run in the replace mode, the entire contents of the "iidatabase" table are destroyed and replaced. All directories containing databases must be scanned. Only the directories that are scanned have their databases included in their new table.

Appendix E
Running ODT-DATA under the Network File System

The following appendix assumes that you are familiar with the basics of mounting filesystems in an environment that provides Network File System (NFS) services.

As the ODT-DATA system administrator, you should understand the following issues and read the scenarios below, before installing ODT-DATA products in an NFS environment.

ODT-DATA provides its own concurrency control services to permit multi-user access to database tables. It is important to know that concurrency control is bound to the shared memory associated with a single processor. The ODT-DATA DBMS lock manager will provide concurrency control for multiple users running on a single processor, but there is no provision within the DBMS server process (iidbms) for synchronized locking of file resources between multiple network nodes.

If ODT-DATA data resides on NFS disk partitions that are exported to other network nodes capable of running the ODT-DATA DBMS server process, transaction concurrency can be compromised. In this environment, the potential exists for two or more ODT-DATA DBMS server processes, using different shared memory, to access the same database independently. Then, the corruption of the user tables and system catalogs is possible. To prevent this, use one of the following examples as a guideline for installing ODT-DATA on multiple nodes that share Network File Systems.

Configuration Scenarios

The following two scenarios have been described to assist the ODT-DATA system administrator at configuration startup time. Each is typical of an ODT-DATA installation in an environment that provides NFS services.

NOTE: The ODT-DATA environment variable $II_SYSTEM is used throughout these scenarios to describe a UNIX file system path specification. When specifying the disk partitions to be exported, mounted, and so on, you will need to replace $II_SYSTEM with the physical path.

Scenario 1

This is an example of a shared ODT-DATA installation, where the ODT-DATA processing is shared by the NFS server and clients using ODT-DATA/NET. The ODT-DATA frontend tools and applications may be executed on both the server node and the client nodes. The DBMS server and other installation related processes are executed only on the server node.

The ODT-DATA system software distribution is installed on the NFS server. The binaries, libraries, and auxiliary files are shared. The $II_DATABASE, $II_CHECKPOINT, and $II_JOURNAL locations are shared. All ODT-DATA DBMS server processes (**iidbms**), the archiver (**dmfacp**) and recovery (**dmfrcp**) processes are executed on the NFS server. Concurrency management is maintained only by the NFS server node. The clients' ODT-DATA frontend application processes are connected to a DBMS server process, on the NFS server, by using ODT-DATA/NET.

NOTE: It is important that the resolution of $II_SYSTEM is the same for the NFS server as for the clients. You can do this with symbolic links.

Server characteristics:

- $II_SYSTEM/ingres is exported to the frontend clients.
- The **iibuild** procedure is executed at ODT-DATA installation time. See details in Chapter 4 of this book.
- The **iidbms, dmfacp,** and **dmfrcp** processes are running.
- The **iistartup** command is in */etc/rc.local*, and it is executed at system startup time.
- ODT-DATA frontend tools and applications may be executed locally on the NFS server node. For example:

```
$  qbf  dbname
```

Client characteristics:

- Only ODT-DATA frontend tools and applications execute on this node. No DBMS server, archiver, or recovery process, or concurrency control is running on this node.
- **$II_SYSTEM/ingres** is imported from the NFS server, to a "mount" location.

- In the following example of a filesystem, configuration for clients, "/install/61" is $II_SYSTEM for both NFS server and clients.

```
$ df
Filesystem kbytes used avail capacity Mounted on
server:/export/root/client
server:/export/exec/odt/usr
server:/install/61/ingres
```

- ODT-DATA frontend tools and applications are connected to the remote DBMS server process (iidbms) on the NFS server node by using ODT-DATA/NET. For example:

```
$ qbf nodename::dbname
```

NOTE: Not all ODT-DATA commands and utilities are available with ODT-DATA frontends executing over ODT-DATA/NET.

Scenario 2

This is an example with separate ODT-DATA installations for the NFS server and clients. The NFS server and clients maintain logically separate installations with distinct $II_DATABASE, $II_CHECKPOINT, and $II_JOURNAL locations. The ODT-DATA system software distribution is installed on the NFS server and can be reinstated entirely on the clients, where symbolic links to the server installation are available. This minimizes disk space usage at the expense of communications traffic overhead.

Clients can share static ODT-DATA files with the NFS server, like the ODT-DATA binaries, libraries, utilities, and technical notes. ODT-DATA DBMS server processes (iidbms), archiver (dmfacp), and recovery (dmfrcp) processes and concurrency control, run on BOTH the NFS server and clients. ODT-DATA frontend tools and applications can be executed either locally or in connection with a remote DBMS server process by using ODT-DATA/NET.

NOTE: To improve performance in the development environment, all clients should have their own copies of the *INGRES/lib* directory.

Server characteristics:

- **$II_SYSTEM/ingres** is exported to NFS clients. For example, to see the list currently exported, type:

```
$ cat /etc/exports
```

- The **iibuild** procedure is executed at ODT-DATA installation time. See details in Chapter 4 of this book.
- The **iidbms, dmfacp,** and **dmfrcp** processes are running.
- The **iistartup** command in */etc/rc.local* is executed at system startup time.
- ODT-DATA frontend tools and applications may be developed locally on the NFS server. For example:

```
$ qbf dbname
```

Client characteristics:

NOTE: The clients will share the *bin, lib, utility,* and *notes* directories with the NFS server, using an NFS mount and symbolic link. The *data, jnl, ckp,* and *files* directories will be set locally using the **iibuild** utility. Create the *files* directory by copying the central files directory.

- **$II_SYSTEM/ingres** is imported from the NFS server, to some "mount" location. This "mount" location will not be used as $II_SYSTEM; a symbolic link will need to be set up for this purpose.
- The **iibuild** command is executed at ODT-DATA installation time. See the filesystem configuration example below, and details in Chapter 4 of this book.
- Concurrency control and **iidbms, dmfacp,** and **dmfrcp** processes are running on the clients.
- The **iistartup** command in */etc/rc.local* is executed at system startup time.
- ODT-DATA frontend applications and tools may be connected to a remote DBMS server process (**iidbms**) by using ODT-DATA/NET:

```
$ qbf nodename::dbname
```

They may also be connected to a local DBMS server process (iidbms):

```
$  qbf  dbname
```

An example of how you configure a filesystem for a client follows:

```
$  df
Filesystem kbytes used avail capacity Mounted on
server:/dev/sd0a/db
server:/dev/sd0b/bck
server:/export/root/client/
server:/export/exec/odt/usr
server:/usr/m/server/usr
server:/install/ingres/m/server/ingres
```

- Identify a location for $II_SYSTEM on the client. Define a symbolic link for the following directories from the mounted location to the $II_SYSTEM location. For this example $II_SYSTEM is set to "/install/61".

```
$ ln -s /m/server/ingres/bin/install/61/ingres/bin
$ ln -s /m/server/ingres/notes/install/61/ingres/notes
$ ln -s /m/server/ingres/lib/install/61/ingres/lib
$ ln -s /m/server/ingres/utility/install/61/ingres/utility
$ ln -s /m/server/ingres/vec/install/61/ingres/vec
$ ln -s /m/server/ingres/dbtmplt/install/61/ingres/dbtmplt
$ ln -s /m/server/ingres/release.doc/install/61/ingres/release.doc
```

- Copy the $II_SYSTEM/files directory from the server node to $II_SYSTEM location of the client node.

```
$  rcp -r server:/m/server/ingres/files/install/61/ingres
```

- Before **iibuild** is executed, a *.version* file must be copied from the server node to the **$II_SYSTEM/ingres** directory of the client. This prevents the entire distribution from being read onto the client nodes during the installation procedure.

```
$  cp  /m/server/ingres/.version/install/61/ingres
```

- Execute installation procedure in Chapter 4, to create directories and set up the installation.

Warning: A version file must exist before **iibuild** is executed.

The following environment variables must be set to directories which are physically resident on the local disk, and NOT set to directories which are resident on an NFS mounted disk.

Example:

```
$II_DATABASE = /db
$II_CHECKPOINT = /bck
$II_JOURNAL = /bck
$II_LOG_FILE = /bck
```

Likewise, alternative ODT-DATA locations must reside on local disks.

NOTE: When upgrading a Scenario 2 installation, copy new file versions to the non-linked directories. Backup copies of the installation-dependent files should be made, as with the *INGRES/files* directory.

Glossary

abstract datatype A datatype that is not native to the operating system, but is implemented with a data structure and a set of operators. Examples of data operators are the interval function for the date and the addition for money.

aggregate A Computation that operates on set of values(for example an average). See also **set function.**

attribute In VIFRED, a characteristic such as highlighting, or, a validation check that affects the display and behavior of a field.

backend The process responsible for interacting with the data. Also called data manager. The backend receives query language commands from a frontend and returns the appropriate data. See also **frontend.**

base table A physical table as opposed to a view. Also used in contrast to a secondary index. See also **table.**

break column A column in a report for which a special action, such as a subtotal, occurs when data values change.

Btree A storage structure characterized by a dynamic index tree.

catalog A table that keeps track of database objects. Catalogs are automatically supplied and maintained by ODT-DATA.

cell The intersection of a row and a column in a table field (or more rarely, in a table).

checkpoint A static backup ODT-DATA creates of a database.

column A vertical selection of data in a table or afield that represents one piece of information.

correlation name An alternate name in SQL for a table, usually a shortened form of the name. See also **range variable.**

data manager The process responsible for interacting with the data. Also called backend. See also **backend.**

data window The area of a form where information can be entered.

database In the relational database model, a collection of tables.

database administrator The ODT-DATA user that owns the database (for example the user that created it).

data set The set of records retrieved by the query statement. In particular, the set of records associated with a table field by a query.

DBA Database administrator.

default A selection provided automatically by ODT-DATA, such as a form in QBF or a default report.

dynamic SQL A part of embedded SQL allowing the user to build queries at runtime.

Embedded SQL An ODT-DATA application development tool in which SQL commands are placed in a third generation language program such as FORTRAN or COBOL.

field An area of a form used for data entry and retrieval, composed of a title, data window and attributes.Also used as a synonym for column. See also **column, table field, simple field.**

form The computerized equivalent of a paper form, where users can enter, store and retrieve data.

frame A piece of an application composed of a form and a menu.

frontend A user interface to ODT-DATA. It can be an ODT-DATA tool (for example QBF) or a custom application. See also **backend.**

FRS (Forms Run-Time System) The part of ODT-DATA that controls the display of forms and user's manipulation of forms and menus.

frskey A logical key used in ODT-DATA application code to refer to a keyboard key.

hash A storage structure characterized by a number of "buckets" (primary pages) where records are placed according to the value of a random function applied to their keys.

heap A default storage structure having no index and no ordering.

inconsistent database A database on which a transaction was not completed. A transaction underway when the system crashed.

index See also **secondary index.**

integrity A backend test to assure that data matches certain specifications.

ISAM (Indexed Sequential Access Method) A storage structure characterized by a static index tree.

JoinDef One or more tables joined together in QBF and functioning as a single object in QBF for data manipulation purposes.

journal A log of transactions since the last static backup. For each transaction, ODT-DATA journals show the change, the date and time of the change, and the user who made it.

key The part of a record that uniquely identifies it (logical key). The column(s) of a table on which a storage structure is built. A heap storage structure has no key.

locking The mechanism by which ODT-DATA makes a multiuser environment possible (for example it protects each user's work from corruption by other users).

Menu key The key that moves the cursor to the menu line.

MST (Multi-statement Transaction) A transaction including several statements identified and executed as a block.

null value A special value representing missing or unknown information.

object Any database entity: table, form, QBF name, application, joindef, graph, report.

ODT-DATA/MENU The tool allowing users to access all of ODT-DATA's menus.

optimizer The part of ODT-DATA responsible for finding the fastest way to execute queries.

page An ODT-DATA page is a 2,048-byte structure with 2,010 bytes available for storing user data.

PERMIT A backend statement to allow users other than the DBA to access data.

QBF (Query By Form) An ODT-DATA tool performing Query Execution; appending, retrieving or modifying data in tables, and, **Join Definition**; specifying a set of tables for a query.

QBF name Mapping of a form to a table or a joindef.

query A data statement for viewing, changing, adding or deleting data.

query target An object used in QBF. Query targets include tables, joindefs and QBF names.

range variable An alternate name for a table, usually a shortened form of the name.

RBF (Report By Forms) The ODT-DATA menu based editor for customizing reports.

record A set of related data in a table; a tuple.Also used to refer to a tuple in the dataset associated with a table field (tuples in the data field are called rows). See also **row**.

relation The technical word for table. See also **table**.

report Displaying information from the database in an easy to use format.

REPORT The ODT-DATA tool for running default or user defined reports.

row A set of related data in a table; a tuple.Also used to refer to a tuple in a table field (tuples in the dataset associated with the table field are called records). See also **record**.

secondary index A table composed of a key and a pointer to the records of the base table. ODT-DATA automatically maintains the index as records are added or updated in the base table.

server A process which provides particular services to a number of processes. The data manager is a server in ODT-DATA release 6.0.

set functions In SQL, a computation that operates on a set of values (for example an average). See also **aggregate.**

simple field A field containing a single piece of data. See also **table field**.

SQL (Structured Query Language) A language used to define, manipulate and protect data.

storage structure A method of arranging the pages of a table. ODT-DATA supports four storage structures: heap, hash, ISAM and Btree.

subquery A SQL subselect nestled within another SQL statement.

subselect A SQL select statement containing only one select keyword without a union or an order by. a subselect issued to build a search condition for the main query.

system catalog See **catalog**.

table A set of data arranged in rows and columns.

table field A field containing several pieces of data (one or more columns).

TABLES A forms based tool accessible from the ODT-DATA/MENU for creating, examining, and deleting tables.

Terminal Monitor An interface where the user can enter query language commands. The two terminal monitors are the line terminal monitor and the full-screen terminal monitor.

title A character string used to identify a field on a form.

transaction A set of statements that function as a unit.Either all or none are executed. A transaction may consist of a single statement or several statements grouped in an MST (Multi-statement transaction).

trim A string of characters on a form used to decorate or instruct.

tuple A row or record in a table.

validation check In VIFRED, a test to insure that data entered into a field meets certain specifications.

view A logical definition of data taken from one or more tables.

VIFRED (Visual Forms Editor) The ODT-DATA menu based editor for customizing forms.

VIGRAPH The ODT-DATA tool for creating, modifying, and running graphs.

Index

This index locates entries by book name and page number. Each of the book names in this volume is indicated with an abbreviation, listed in the table below. A key with this information is at the bottom of each following index page.

Book Name	Abbreviation
Administering ODT-VIEW	*VIEW*
Administering ODT-OS	*OS*
Administering ODT-NET	*NET*
Administering ODT-DOS	*DOS*
Administering ODT-DATA	*DATA*

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

12-12-89

104-000-930
26179

104-000-930 12/89
26179