SCO

SCO OpenServer™ System Administration Guide

SCO OpenServer™

SCO OpenServer™
System Administration Guide

Document Version: 5.0.4
30 May 1997

About this book 1

Chapter 1
Administering user accounts 7

Chapter 5

Maintaining system security 223

About this book

The *System Administration Guide* describes tasks related to the base SCO operating system — procedures the administrator must perform to ensure smooth and efficient operation of all services.

NOTE Your SCO OpenServer™ system requires monitoring of its operation and a regular schedule of maintenance. Refer to Chapter 9, "Administering SCO systems" in the *SCO OpenServer Handbook* for information on how to approach system administration.

You will find the information you need more quickly by reviewing:

- "How this book is organized" (this page)
- "Related documentation" (page 2)
- "Typographical conventions" (page 5)

Although we try to present information in the most useful way, you are the ultimate judge of how well we succeed. Please let us know how we can improve this book (page 6).

How this book is organized

This book explains how to:

- manage user accounts, including passwords, restrictions, and security defaults (page 7)
- set up and mantain filesystems (page 53)
- prevent data loss by backing up filesystems (page 105)
- maintain printers and control print services (page 141)

- understand and monitor system security (page 223)
- configure and maintain audit records of system operations (page 259)
- set up connections to remote systems over phone lines (page 295)
- improve system performance using disk arrays, including high-performance and fault-tolerant RAID configurations (page 355)

Appendices provide additional information about tailoring system startup (page 401), non-graphical console and terminal features (page 411), the layout of system directories (page 423), and analyzing a memory dump from a system failure (page 429).

C2 security documentation requirements

The C2 requirements for security documentation (the *Trusted Facilities Manual* and *Security Features User's Guide*) are satisfied by the following:

- Chapter 1, "Administering user accounts" (page 7)
- Chapter 5, "Maintaining system security" (page 223)
- Chapter 9, "Using a secure system" in the *Operating System User's Guide*

Related documentation

SCO OpenServer systems include comprehensive documentation. Depending on which SCO OpenServer system you have, the following books are available in online and/or printed form. Access online books by double-clicking on the Desktop **Help** icon. Additional printed versions of the books are also available. The Desktop and most SCO OpenServer programs and utilities are linked to extensive context-sensitive help, which in turn is linked to relevant sections in the online versions of the following books. See "Getting help" in the *SCO OpenServer Handbook*.

NOTE When you upgrade or supplement your SCO OpenServer software, you might also install online documentation that is more current than the printed books that came with the original system. For the most up-to-date information, check the online documentation.

Release Notes
: contain important late-breaking information about installation, hardware requirements, and known limitations. The *Release Notes* also highlight the new features added for this release.

SCO OpenServer Handbook
provides the information needed to get your SCO OpenServer system up and running, including installation and configuration instructions, and introductions to the Desktop, online documentation, system administration, and troubleshooting.

Graphical Environment Reference
contains the manual pages for the X server (section X), the Desktop, and X clients from SCO and MIT (section XC).

Guide to Gateways for LAN Servers
describes how to set up SCO® Gateway for NetWare® and LAN Manager Client software on an SCO OpenServer system to access printers, filesystems, and other services provided by servers running Novell® NetWare® and by servers running LAN Manager over DOS, OS/2®, or UNIX systems. This book contains the manual pages for LAN Manager Client commands (section LMC).

Mail and Messaging Guide
describes how to configure and administer your mail system. Topics include **sendmail**, MMDF, **SCO Shell Mail**, **mailx**, and the Post Office Protocol (POP) server.

Networking Guide
provides information on configuring and administering TCP/IP, NFS®, and IPX/SPX™ software to provide networked and distributed functionality, including system and network management, applications support, and file, name, and time services.

Networking Reference
contains the command, file, protocol, and utility manual pages for the IPX/SPX (section PADM), NFS (sections NADM, NC, and NF), and TCP/IP (sections ADMN, ADMP, SFF, and TC) networking software.

Operating System Administrator's Reference
contains the manual pages for system administration commands and utilities (section ADM), system file formats (section F), hardware-specific information (section HW), miscellaneous commands (section M), and SCO Visual Tcl™ commands (section TCL).

Operating System Tutorial
provides a basic introduction to the SCO OpenServer operating system. This book can also be used as a refresher course or a quick-reference guide. Each chapter is a self-contained lesson designed to give hands-on experience using the SCO OpenServer operating system.

Operating System User's Guide
provides an introduction to SCO OpenServer command-line utilities, the SCO Shell utilities, working with files and directories, editing files with the **vi** editor, transferring files to disks and tape, using DOS disks and files

in the SCO OpenServer environment, managing processes, shell programming, regular expressions, **awk**, and **sed**.

Operating System User's Reference
contains the manual pages for user-accessible operating system commands and utilities (section C).

PC-Interface Guide
describes how to set up PC-Interface™ software on an SCO OpenServer system to provide print, file, and terminal emulation services to computers running PC-Interface client software under DOS or Microsoft® Windows™.

Performance Guide
describes performance tuning for uniprocessor, multiprocessor, and networked systems, including those with TCP/IP, NFS, and X clients. This book discusses how the various subsystems function, possible performance constraints due to hardware limitations, and optimizing system configuration for various uses. Concepts and strategies are illustrated with case studies.

SCO Merge User's Guide
describes how to use and configure an SCO® Merge™ system. Topics include installing Windows, installing DOS and Windows applications, using DOS with the SCO OpenServer operating system, configuring hardware and software resources, and using SCO Merge in an international environment.

SCO Wabi User's Guide
describes how to use SCO® Wabi™ software to run Windows 3.1 applications on the SCO OpenServer operating system. Topics include installing the SCO Wabi software, setting up drives, configuring ports, managing printing operations, and installing and running applications.

The SCO OpenServer Development System includes extensive documentation of application development issues and tools.

Many other useful publications about SCO systems by independent authors are available from technical bookstores.

Typographical conventions

This publication presents commands, filenames, keystrokes, and other special elements as shown here:

Example:	Used for:
lp or **lp**(C)	commands, device drivers, programs, and utilities (names, icons, or windows); the letter in parentheses indicates the reference manual section in which the command, driver, program, or utility is documented
/new/client.list	files, directories, and desktops (names, icons, or windows)
root	system, network, or user names
filename	placeholders (replace with appropriate name or value)
⟨Esc⟩	keyboard keys
`Exit program?`	system output (prompts, messages)
yes or **yes**	user input
"Description"	field names or column headings (on screen or in database)
Cancel	button names
Edit	menu names
Copy	menu items
File ➪ Find ➪ Text	sequences of menus and menu items
open or **open**(S)	library routines, system calls, kernel functions, C keywords; the letter in parentheses indicates the reference manual section in which the file is documented
$HOME	environment or shell variables
SIGHUP	named constants or signals
"adm3a"	data values
employees	database names
orders	database tables
`buf`	C program structures
`b_b.errno`	structure members

How can we improve this book?

What did you find particularly helpful in this book? Are there mistakes in this book? Could it be organized more usefully? Did we leave out information you need or include unnecessary material? If so, please tell us.

To help us implement your suggestions, include relevant details, such as book title, section name, page number, and system component. We would appreciate information on how to contact you in case we need additional explanation.

To contact us, use the card at the back of the *SCO OpenServer Handbook* or write to us at:

Technical Publications
Attn: CFT
The Santa Cruz Operation, Inc.
PO Box 1900
Santa Cruz, California 95061-9969
USA

or e-mail us at:

techpubs@sco.com or ... *uunet!sco!techpubs*

Thank you.

Chapter 1

Administering user accounts

User accounts help the system administrator keep track of the people using the system and control their access to system resources. Accounts also help organize user files and control their access by other users.

There are several aspects to account administration:

- Users — adding, modifying (page 10), and removing (page 12)
- Groups — adding, modifying, and removing (page 19)
- Passwords — assigning and controlling (page 22)
- Logins — controlling locks and login limits (page 29)
- Powers — assigning superuser powers (page 32)
- Security — changing the system security profile (page 41)
- Troubleshooting — solving problems with the **Account Manager** (page 49)

The Account Manager interface (this page) provides a convenient, interactive way to perform most account administration tasks.

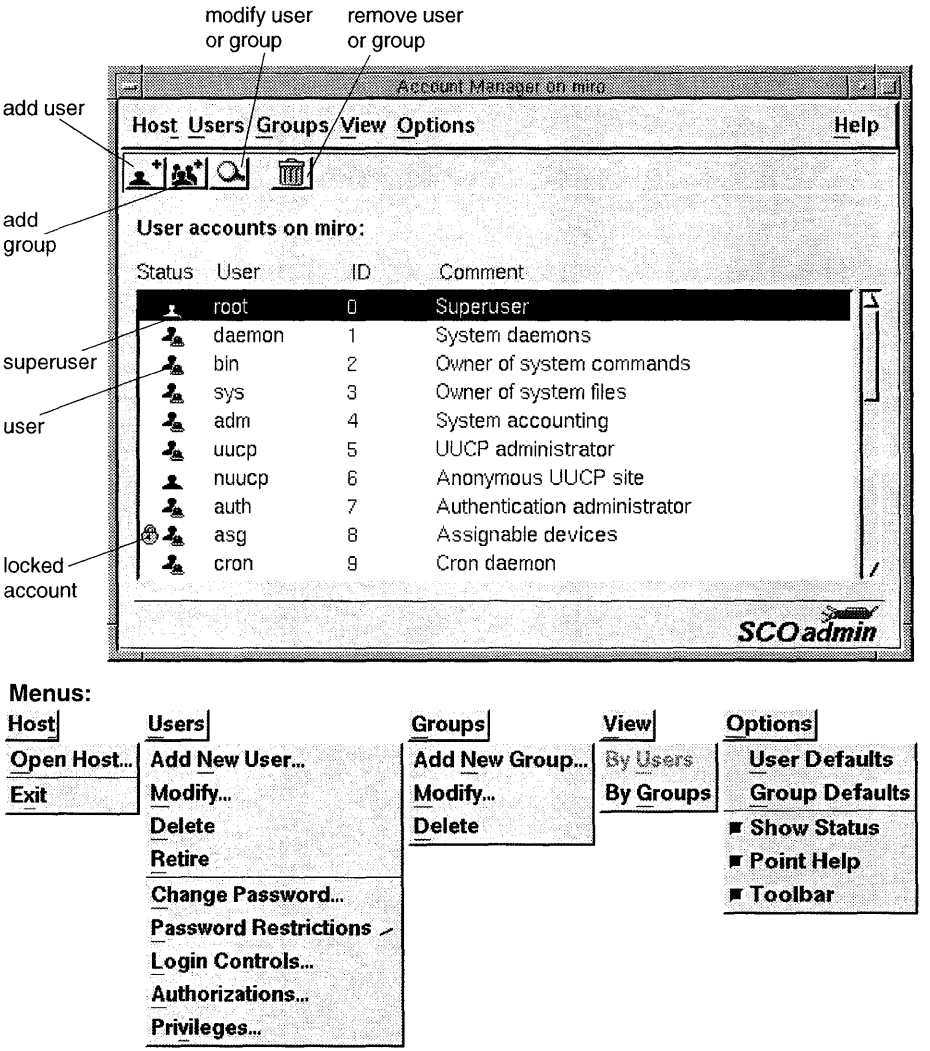
The Account Manager interface

The **Account Manager** allows you to create or modify user accounts. You can start the **Account Manager** in any of these ways:

- Double-click on the **Account Manager** icon in the *System Administration* window on the Desktop.
- Start the SCOadmin launcher by entering **scoadmin** on the command line, then selecting **Account Manager**.

- Enter **scoadmin account manager** on the command line (or abbreviate to **scoadmin a**).

For more information on using SCOadmin managers, see "Administering your system with SCOadmin" in the *SCO OpenServer Handbook*.



The main display of the **Account Manager** includes a list of accounts on the system. The icons (or characters) in the "Status" column indicate the type of account and its status. By default, the Status information is not displayed to keep startup time to a minimum. To display status information, select **Show Status** from the **Options** menu.

WARNING If you have a large number of accounts on the system, the collection of status information takes a long time and will delay startup of the **Account Manager**. We recommend not using the **Show Status** selection under these circumstances.

You can display a list of groups on the system by selecting **Groups** from the **View** menu.

Description	Graphical	Character
distributed account		>
group		G
locked account		#
account has no password		!
pseudouser		P
retired user		R
superuser	 (red)	S
user	 (black)	*none*

Figure 1-1 Account Manager status icons

Authorization

When run from an account other than *root*, use of the **Account Manager** requires the **auth** authorization. See "Assigning subsystem authorizations" (page 32) for more information.

About default selections

Many selections of the **Account Manager** have multiple choices: **Yes** and **No** (or a single numerical value), and **Default**. When you choose the **Default** option, you are not simply using a static value that is copied into the user's

account. Instead, you are configuring the account to use a default value that can change *dynamically*. That is, you can change the system default value that immediately affects all accounts. With this in mind, you can configure user accounts to use either a specific static value, or a dynamic one associated with most users.

Adding and modifying user accounts

In the **Account Manager**, select **Add New User** from the **Users** menu, or select a user from the list and select **Modify** from the **Users** menu.

When adding a new user, you need only enter a name and assign a password. If you select **Set password later**, the user will not be able to log in until you assign one as described in "Setting or changing a user password" (page 22).

You can alter any of these attributes (system defaults are used if you do not):

- User ID number (page 13)
- Comment (optional)
- Password (page 22)
- Login shell (page 16)
- Distribution (local or NIS) as described in "Managing distributed user accounts" in the *Networking Guide*
- Home directory (page 17)
- Login group (page 14)
- Group membership (page 15)
- Login controls and locks (page 29)
- Powers — including authorizations and privileges (page 32)
- User type (page 17)
- Priority of the user's processes (page 18)

NOTE Remote administration (using the **Open Host** selection of the **Host** menu) requires user equivalence on the machine you plan to administer. The remote machine must recognize the account being used to administer users on the local machine. See "Adding user equivalence" in the *Networking Guide* for more information. As on the local machine, non-*root* accounts require the **auth** authorization to run the **Account Manager**. See "Assigning subsystem authorizations" (page 32) for more information.

To change the system default values, select **User Defaults** from the **Options** menu.

You can also create users on the command line:

useradd *username*

Default values are used unless overridden by command line options described in the **useradd**(ADM) manual page.

See also:

- "Removing or retiring a user account" (page 12)
- "Copying user accounts" (page 47)
- "Using account templates" (this page)

Using account templates

Account templates are files that contain a list of account attributes (such as group membership or privileges) for use with the account command line utilities. Templates make it easier to specify attributes that would otherwise require a long command line.

NOTE Templates do not work with the **Account Manager**. They are are designed for use with command line utilities only.

Here is an example template, *template1*, that specifies several attributes, including group membership, kernel privileges, login restrictions, password restrictions, and authorizations:

```
{ groups { gedemondan czillian dillian type41 oolakash } }
{ kernelAuths { chown execsuid su } }
{ maxLoginAttempts 8 }
{ passwdExpirationTime 60 }
{ passwdLifetime 120 }
{ passwdRunGenerator 1 }
{ subsystemAuths { printerstat printqueue queryspace } }
```

Templates can be used to create new accounts, modify existing accounts, or change the system defaults.

Here is an example that creates a new account for user *mavrac* that uses *template1*:

useradd -X template1 mavrac

The new account will use all the default account values except for those listed in the template file. To modify an existing account for the same user, you would simply substitute **usermod** for **useradd**.

You can also modify account defaults using a template file. Assuming you wanted to replace the current system defaults with the values in *template1*, you would use this command:

usermod -D -X template1

You can use the **userls** command to list account attributes that you can in turn use to create a template. For example, this command directs the attributes for user *mavrac* into a file:

userls -l mavrac -A > template1

You must edit the output from **userls** because it contains all attributes, including status information that cannot be changed (such as last login time), but you can us it as a basis for a template file.

See also:

- **userls**(ADM) manual page
- **groupadd**(ADM) manual page
- **groupls**(ADM) manual page

Removing or retiring a user account

In the **Account Manager**, select a user name, then select **Delete** or **Retire** from the **Users** menu. If you are operating with the Improved or High security profiles, only **Retire** is available.

Removing or retiring a user account does not remove the user's files; the system administrator must do this manually by deleting the directory. In addition, there may be other files that belong to the user. See "Changing ownership of files with an obsolete UID/GID" (page 14) for more information.

NOTE When operating with the High (C2) security profile, a user is never removed from the system. Once assigned, a UID is never reused. Instead, a user account is "retired," or removed from service. Under other profiles, a user can be completely removed from the system. To allow UIDs to be reused regardless of the security profile, you can define **REUSEID=YES** in */etc/default/login*.

See also:

- "Reactivating a retired user account" (page 13)
- **rmuser**(ADM) manual page (for command line interface)
- **unretire**(ADM) manual page (for command line interface)

Reactivating a retired user account

Use the **unretire**(ADM) command, substituting the account name for *username*:

unretire *username*

The user should then be able to log in to the account.

NOTE When operating under the High (C2) security profile, a user is never removed from the system. Therefore **unretire** will not function under the High security profile.

Setting and changing user and group IDs (UID/GID)

Each user and group is assigned an identification number (UID or GID). This ID number is stamped on all files, directories, and processes on local and NFS-mounted filesystems. When you create a new user or group, a new UID or GID number is automatically assigned, but you can specify another by entering it in the text field.

WARNING Within a network environment, each user and group must have a unique identification number across the entire network, rather than just on the home machine. For more information, see "Establishing user equivalence" in the *Networking Guide* and Chapter 7, "Configuring the Network Information Service (NIS)" in the *Networking Guide*.

To change the ID number for an existing user or group, use the **usermod**(ADM) or **groupmod**(ADM) command. Changing an ID number of a user or group does not change the ID on files owned by the user or group; the system administrator must do this manually as described in "Changing ownership of files with an obsolete UID/GID" (page 14).

To alter the range of UIDs from which you select for new users, select **User Defaults** from the **Options** menu. To alter the range of GIDs from which you select for new groups, **Group Defaults** from the **Options** menu.

NOTE When operating under the High security profile, a UID cannot be changed. Under other profiles, UIDs can be changed without restriction.

See also:

- **usermod**(ADM) manual page (for command line interface)
- **groupmod**(ADM) manual page (for command line interface)
- "Managing groups" (page 19)

- "Setting the group ID for file creation in a directory" (page 21)

Changing ownership of files with an obsolete UID/GID

If you have changed or removed a UID or GID, you must change the ownership of their files and check your filesystems for orphaned files. Files without a real owner have a number in the owner and/or group name fields:

```
-rw-r--r--   1 obie     pub        68476 Nov 16 12:06 accts.s
-rw-r--r--   1 15625    pub          508 Oct 31 11:15 balance
-rw-r--r--   1 obie     pub        40596 Aug 31 13:19 report.2
```

In this example, the file *balance* is an orphaned file. The number appears because files are stamped with the ID number rather than the user or group name.

Use the **find**(C) utility to locate and change the ownership of files. This command line finds all files on the system owned by user ***UID*** and changes ownership to user *newowner*:

find / -user ***UID*** **-print | xargs -t chown** *newowner*

This variation changes the group ownership:

find / -user ***GID*** **-print | xargs -t chgrp** *newgroup*

NOTE These examples assume a search of the entire system (including all mounted filesystems whether local or imported). To restrict the search to a single filesystem, use the pathname instead of /. In addition, you can skip mounted filesystems by including the **-mount** option, or restrict the search to local filesystems with the **-local** option.

Instead of changing the ownership, you can perform other actions, such as archiving the files; see "Locating files" (page 88).

See also:

- **find**(C) manual page
- **xargs**(C) manual page

Changing user login groups

In the **Account Manager**, select a user name, then select **Modify** from the **Users** menu. Use the **Change Group Membership** button to change the value in the "Login Group" field.

The login group is the default group to which the user belongs. Files and directories created by the user are owned by this group. Users can temporarily change their login group using the **sg**(C) command.

NOTE If you add a user to a group that does not exist, you are given the opportunity to create it. When a user's login group is set this way, the group will not be added to the "Member of" column. This is normal.

To change the system default login group, select **User Defaults** from the **Options** menu.

See also:

- "Managing groups" (page 19)
- "Changing a user's group membership" (this page)
- **usermod**(ADM) manual page (for command line interface)

Changing a user's group membership

In the **Account Manager**, select a user name, then select **Modify** from the **Users** menu. Click on the **Change Group Membership** button.

To add user to a group, select an entry in the "Other Groups" column and click on the **Add** button.

To remove a user from a group, select an entry in the "Member of" column and click on the **Remove** button.

You can search for a specified group by entering the name in the "Search for:" field.

To change the set of default groups assigned to new users, select **User Defaults** from the **Options** menu.

NOTE There is a limit to the number of groups a user can be a member of at one time. See "Changing the limit on simultaneous group membership" (page 21) for more information.

See also:

- "Managing groups" (page 19)
- "Changing user login groups" (page 14)
- **usermod**(ADM) manual page (for command line interface)

Changing user login shells

In the **Account Manager**, select a user name, then select **Modify** from the **Users** menu. Use the **Login Shell** button to select from the available login shells (this page).

To change the system default login shell, select **User Defaults** from the **Options** menu.

Each shell has one or more environment files specific to that shell (for example, *.profile* and *.kshrc* for **ksh**(C)). To add environment files to the home directory, select **Add Shell Environment Files to Home Directory**. (This button has no effect on a newly created home directory; environment files are always added in this case.)

See also:

- "What happens when you log in" in the *Operating System User's Guide*
- Appendix D, "Sample shell startup files" in the *Operating System User's Guide*
- **usermod**(ADM) manual page (for command line interface)
- "Adding and modifying default environment files" (page 19)

Login shells

These shells are available:

Bourne shell (*/bin/sh*)
: First shell to be developed; includes wildcards, basic command language. Also available in a restricted version (page 17).

C shell (*/bin/csh*)
: Different language syntax from Bourne and Korn shell family (similar to the C programming language). Includes command history recall (reuse of recently issued commands without retyping them), aliases (defining alternative names for commands), and limited ability to redirect input and output.

Korn shell (*/bin/ksh*)
: Compatible superset of Bourne shell facilities; includes command history editing (editing and reissuing previously typed commands interactively) and aliases. In addition, supports job control (manipulation of background processes), and extended language syntax. Recommended shell. Also available in a restricted version (page 17).

SCO shell (*scosh*)
: Menu-based shell (character interface).

uucp Non-interactive shell for UUCP login accounts that use **uucico**(ADM). Do not assign this to a normal user account. See "Creating login accounts for sites dialing in" (page 302).

See also:

- Chapter 10, "Configuring and working with the shells" in the *Operating System User's Guide*

Restricted shells

There are restricted versions of the Bourne and Korn shells (**rsh** and **rksh**) that prohibit changing directory with **cd**, setting the value of **$PATH**, using command names containing slashes, and redirecting output using > and >>.

Changing user home directories

In the **Account Manager**, select the user name, then select **Modify** from the **Users** menu. Use the **Change Home Directory** button to display the home directory options (only the first three are available when you add a new user):

Home Directory or **Home Directory Base Path**
allows you to enter the home directory name. When changing user defaults, the base path is the directory where all home directories will reside (example: */usr*).

Home Directory Permissions
allows other users in the same group or any others to access the user's home directory. This option is only available when creating a new directory or modifying the default location of user directories.

Create Home Directory
adds the new home directory to the system. Deselect this option if you wish to use an existing directory.

Move Files from Old Home Directory
moves all the user's existing files to the new home directory.

To change the system default location, select **User Defaults** from the **Options** menu.

NOTE Changing the system default location for home directories does not move existing accounts; only new accounts are placed in the new location.

Changing user type

Optional user type labels are provided for sites wishing to distinguish between different types of users.

NOTE Retired accounts have the type of "retired," so if you set the type to this value the account is promptly retired. Otherwise, user types are merely informational labels. They do not confer any special privileges. The user type attribute is not accessible from the **Account Manager**.

To change the type for an account, use this command line:

usermod -x "{userType *type*}" *user*

where ***user*** s the name of the account and ***type*** is one of:

root
operator
sso
admin
pseudo
general
retired

To change the system default used for account creation, use this command line:

usermod -D -x "{userType *type*}"

By default user accounts have the type label of "general", and you need not change it. Anonymous accounts like *sysinfo* and *uucp* have the label "pseudo-user". Each pseudo-user has an accountable user, who is considered responsible for that account. (For example, *root*, an individual, is defined as the accountable user for all pseudo-user accounts.)

Changing user priority

The priority setting determines the scheduling priority for the user's processes: the greater the value, the higher the priority.

To change the priority for an account, use this command line:

usermod -x "{nice *value*}" *username*

Processes using traditional scheduling have priorities 0 (low) to 127 (high); processes using real-time scheduling have priorities 128 to 255.

To change the system default used for account creation, use this command line:

usermod -D -x "{nice *value*}"

See also:

- **nice**(C) manual page

Adding and modifying default environment files

You can modify the default configuration files that are added to the home directory for a new account. For example, **csh**(C) has prototype *.login* and *.cshrc* files that are installed in a user's directory when **csh** is selected as that user's login shell. Each shell has a directory of these prototype files in */usr/lib/mkuser*.

NOTE There are also system-wide files in */etc* that initialize the Bourne/Korn and C-shell environment. See "Modifying .profile and .login files" (page 409) for more information.

You can also add login shells (and configuration files) to the set of login shells available from the **Account Manager**. Examine the existing files and follow their example. Make sure the permissions and ownership are consistent with these files. When you install the files, the new shell is available in the **Account Manager**.

See also:

- Appendix D, "Sample shell startup files" in the *Operating System User's Guide*

Managing groups

Groups allow you to control access to files and directories by creating work-groups. These tasks are associated with groups:

- Adding or modifying a group (page 20)
- Removing a group (page 21)
- Changing user login groups (page 14)
- Changing a user's group membership (page 15)
- Setting the group ID for file creation in a directory (page 21)
- Changing the limit on simultaneous group membership (page 21)

WARNING Do not change the GIDs of any default system groups.

See also:

- "About groups" (page 20)

About groups

Groups allow sets of users to share files. The group permissions on a file or directory determine whether members of a group can access them.

You can set the group of a file or directory by changing its properties from the Desktop or by using the **chgrp**(C) command. For more information on properties, see "Controlling file permissions" in *Using the Desktop*.

A user can be a member of several groups at once, and can access any file or directory that is within their group set (if group permissions allow). This membership set is known as the supplemental group list. By default, files created by a user have the group ID of their login group. Users can change their current working group (also known as *effective* group ID) or manipulate their supplemental group list with the **sg**(C) command.

NOTE When files are created in a directory, the group ownership is determined by the setting of the GID bit. See "Setting the group ID for file creation in a directory" (page 21).

See also:

- "Access control for files and directories" in the *Operating System User's Guide*
- "Changing the limit on simultaneous group membership" (page 21)

Adding or modifying a group

In the **Account Manager**, select **Add New Group** from the **Groups** menu. Enter the group name. You can change the suggested group ID number by overwriting the text field.

To modify an existing group, select **Group** from the **View** menu, select a group name, then select **Modify** from the **Groups** menu.

To add user to the group, select an entry in the "Other Users" column and click on the **Add** button.

To remove a user from the group, select an entry "Users In Group" column and click on the **Remove** button. You can also search for a user name to select.

To alter the range of GIDs from which you select for new groups, select **Group Defaults** from the **Options** menu.

To create groups on the command line:

groupadd *groupname*

Default values are used unless overridden by command line options — see **groupadd**(ADM).

NOTE You can also edit the file */etc/group* to add or modify groups as desired, but using the **Account Manager** or **groupadd** is recommended.

See also:

- "Removing a group" (this page)

Removing a group

In the **Account Manager**, select **Group** from the **View** menu, then select the group name. Then, select **Delete** from the **Groups** menu.

Removing a group does not change the GID of files on the system. The administrator must do this manually. See "Changing ownership of files with an obsolete UID/GID" (page 14) for more information.

See also:

- "Adding or modifying a group" (page 20)
- **groupdel**(ADM) manual page (for command line interface)

Setting the group ID for file creation in a directory

By default, the GID (group identifier) of a newly created file is set to the GID of the creating process or user. Setting the SGID bit on a directory causes all subsequent new files to take the GID of that directory. This makes it easier for users to share files, because it ensures that all files will have the same GID, even if the users don't have the same login group.

To set the SGID bit on a directory, use this command, substituting the directory name for *directory*:

chmod g+s *directory*

To remove the bit, replace the "+" with a "-" in the **chmod** command.

Changing the limit on simultaneous group membership

The maximum number of groups that a user can be in at one time is 8 by default. There are no warning messages displayed when this number is exceeded, but when the **sg**(C) command is used to extend a user's group list using the verbose option (**-v**), this error message is displayed:

```
sg: unable to add group group as supplemental group list full
```

This limit is controlled by the **NGROUPS** tunable kernel parameter. To change this value, use the **Hardware/Kernel Manager** or the **configure**(ADM) command. Select category 7, "User and group configuration", then change the value of **NGROUPS**. The kernel must then be relinked and booted for the new value to take effect. See "Relinking the kernel" in the *SCO OpenServer Handbook* for more information.

Managing passwords

Passwords are the first line of defense against unauthorized users. Protect your system by requiring passwords for all user accounts.

These tasks are associated with passwords:

- Setting or changing a user password (this page)
- Controlling password expiration (page 23)
- Controlling password selection (page 24) — including the use and generation of passwords, their length and obviousness
- Setting passwords for dial-in lines (page 29)

Setting or changing a user password

In the **Account Manager**, select a user name, then select **Change Password** from the **Users** menu.

You have these options:

Enter a new password
allows you to create a new password.

Keep existing password
retains the current password (this option is not available during account creation).

Use machine generated password
allows you to generate a password for the user.

Remove password
deletes the password and allows the user to log in without one.

Force password change at next login
forces users to change their password the next time they log in.

If you are entering a new password, type it in the "Enter Password" field (the password is not displayed). You are then switched to the "Confirm Password" field to enter it once more.

Use the **Generate a password** button to create a password for a user. Generated passwords are pronounceable, meaning that they are nonsense words rather than just random strings of letters (for example: *juhahiwa*). The password is automatically entered in the password fields; click on **OK** to accept the password, otherwise click on the **Generate...** button until a satisfactory password is generated.

Users can also generate their own passwords, unless you prevent them from running the generator. See "Allowing users to generate passwords" (page 25) for more information.

See also:

- Table 1-1, "Password checking by security profile" (page 26)
- **passwd**(C) manual page (for command line interface)

Controlling password expiration

In the **Account Manager**, select a user name, then select **Password Restrictions** from the **Users** menu, then select **Expiration**.

System defaults are used unless you use the default toggle buttons to unstipple the text fields:

Days allowed between changes
sets how long users must wait before they can change their password again. This prevents users from changing their passwords when they expire and then immediately changing them back to the previous one.

Days until password expires
sets how long a password is valid. When the password expires, the user is prompted to set a new password when they log in.

Days until account is locked
sets the interval between the time when the password expires and the account is automatically locked (preventing the user from logging in). Also known as the "password lifetime".

To change the system default values, use these commands:

usermod -D -x "{passwdMinChangeTime *value*}"
usermod -D -x "{passwdExpirationTime *value*}"
usermod -D -x "{passwdLifeTime *value*}"

You can change the value for an individual user with the **usermod**(ADM) command by omiting the **-D** option and appending the user name to the above commands.

NOTE The Low and Traditional security profiles use password restrictions that are very lenient: passwords do not expire, accounts are not locked, and there is no minimum interval between password changes.

The default account initialization files (*.cshrc, .profile, .kshrc,* and so forth) automatically execute the **prwarn**(C) utility at login time to warn users about impending password expiration.

See also:

- "Locking or unlocking a user account" (page 31)
- **usermod**(ADM) manual page (for command line interface)

Controlling password selection

Password selection constraints give the administrator these capabilities:

- Allowing accounts without passwords (this page)
- Preventing users from changing their passwords (page 25)
- Restricting password obviousness (page 26)
- Allowing users to generate passwords (page 25)
- Setting password length (page 28)

Allowing accounts without passwords

In the **Account Manager**, select a user name, then select **Password Restrictions** from the **Users** menu, then select **Selection**.

To permit a user to log in without a password, set **Password Required** to **No**. Accounts without passwords are a major security risk. To use the system default value (page 9), set it to **Default**.

To change the system default value, use this command line:

usermod -D -x "{passwdNullAllowed *value*}"

where ***value*** is either 1 (no password required) or 0 (a password is required).

You can change the value for an individual user with the **usermod**(ADM) command by omiting the **-D** option and appending the user name to the above command.

WARNING Removing the requirement for passwords does not delete existing passwords. The administrator must change each password as described in "Setting or changing a user password" (page 22) and set the password to blank, or use the **passwd**(C) command line.

Preventing users from changing their passwords

In the **Account Manager**, select a user name, then select **Password Restrictions** from the **Users** menu, then select **Selection**.

Set **User can choose own** to **No**. Users will then have to get passwords from the accounts administrator when their passwords expire, or the password generator will create them. To use the system default value (page 9), set it to **Default**.

To change the system default value, use this command line:

usermod -D -x "{passwdChooseOwn *value*}"

where ***value*** is either 1 (users can choose their own password) or 0 (a password is supplied by the administrator or the password generator).

You can change the value for an individual user with the **usermod**(ADM) command by omiting the **-D** option and appending the user name to the above command.

Allowing users to generate passwords

In the **Account Manager**, select a user name, then select **Password Restrictions** from the **Users** menu, then select **Selection**.

You can choose to have the system 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.

To permit users to generate (but not choose) a new password, set **User can run generator** to **Yes**. To use the system default value (page 9), set it to **Default**.

To change the system default value, use this command line:

usermod -D -x "{passwdRunGenerator *value*}"

where ***value*** is either 1 (the user can run the generator) or 0 (the user cannot).

You can change the value for an individual user with the **usermod**(ADM) command by omiting the **-D** option and appending the user name to the above command.

Restricting password obviousness

An important part of password control is ensuring that passwords are difficult to guess without being too complex to remember. You can prevent users from using passwords that are too easy to guess, like dictionary words or system names.

In the **Account Manager**, select a user name, then select **Password Restrictions** from the **Users** menu, then select **Selection**.

Set **Check for Obviousness** to **Yes** to run complex checks on passwords. The meaning of **Yes** and **No** varies with the security profile level chosen. To use the system default value (page 9), set it to **Default**. The meaning can also be set independent of the security profile as described in "Customizing password checking" (page 27).

To change the system default value, use this command line:

usermod -D -x "{passwdCheckedForObviousness *value*}"

where ***value*** is either 1 (use complex checks) or 0 (use less restrictive checks).

You can change the value for an individual user with the **usermod**(ADM) command by omiting the **-D** option and appending the user name to the above command.

Table 1-1 Password checking by security profile

Security Defaults	Check for Obviousness No	Yes
Low	-	-
Traditional	System V	System V-plus
Improved/High	**goodpw** weak	**goodpw** strong

System V (traditional UNIX System V checking) checks that a password:

- is not a rotation of login name (moving characters from beginning to end of the word and vice versa)
- contains at least 2 alphabetic and 1 non-alphabetic characters
- contains at least 3 characters different from old password
- has a minimum length defined by **PASSLENGTH** in */etc/default/passwd* or, if **PASSLENGTH** is undefined or set to an asterisk (*), pass a special length check based on the delay between login attempts and the password lifetime for that user

System V-plus (System V with additions) checks that a password is:

- not a palindrome
- not a group or user name

goodpw weak checks that a password does:

- not contain the strings "SCO", "XENIX", or "UNIX" (defined in */usr/lib/goodpw/reject*)
- not contain a user, group, machine, or alias name
- pass special length checks based on character mixes (these override the set minimum length and are defined in */usr/lib/goodpw/match*):
  - If a password consists of all alphabetic characters of the same case, it must have a length of at least 6.
  - If the password consists of two types of alphanumeric characters, it must have a length of at least 5.
  - If the password consists only of non-alphanumerics (symbols), or a mixture of uppercase, lowercase and numerics, it must have a length of at least 4.

goodpw strong (**goodpw** weak plus additional checks) checks that a password:

- does not contain a dictionary word
- is not a rotation of a user, group, machine, alias name, or dictionary word

The **goodpw**(ADM) checks are defined in the */etc/default/goodpw* file and supplemented or modified by files in the */usr/lib/goodpw* directory. Refer to "Customizing password checking" (this page) for more information.

NOTE Obviousness checking will prevent certain penetrations based on dictionary checking, but such repeated break-in attempts are better controlled with login limits — see "Setting login restrictions on terminals" (page 30). Obviousness checks increase the time required to change a password.

For information on using the command line interface, see the **usermod**(ADM) manual page.

Customizing password checking

The **goodpw**(ADM) utility also allows you to customize password checking. The file */etc/default/goodpw* contains the password control settings. These settings allow you to specify if passwords are checked against dictionary words, word rotations, and user, group, and system names.

NOTE Password checking can also be set by editing */etc/default/passwd* and changing the value of **GOODPW** as follows:

YES	use **goodpw**
NO	use standard UNIX system checking
NONE	perform no password checking

You can also define regular expressions (character combinations and arrangements) that all passwords must match (or not match) with the files */usr/lib/goodpw/match* and */usr/lib/goodpw/reject*, respectively. See **goodpw**(ADM) for more information.

Setting password length

Password length is controlled by three parameters:

- minimum length
- maximum generated length — the limit on passwords created by the password generator
- number of significant segments — see "Password compatibility across UNIX systems" (page 49)

The maximum length of non-generated passwords is 80 characters.

To reconfigure the minimum length, change the value of **PASSLENGTH** in */etc/default/passwd*. If **PASSLENGTH** is removed from the file or is set to an asterisk (`PASSLENGTH=*`), the value is calculated by the system; see "Restricting password obviousness" (page 26) for more information.

You can configure the generated length for individual users with the **Account Manager**. Select a user name, then select **Password Restrictions** from the **Users** menu, then select **Selection**.

To change the system default value, use this command line:

usermod -D -x "{passwdGeneratedLength *value*}"

where ***value*** has a maximum value of 80.

You can change the value for an individual user with the **usermod**(ADM) command by omiting the **-D** option and appending the user name to the above command.

See also:

- "Allowing users to generate passwords" (page 25)

Setting passwords for dial-in lines

You can define special dial-in passwords on selected tty lines, requiring selected classes of users to enter dial-in passwords. Logging information, including the last time of connection, is stored just as with normal logins.

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*.

Use this command line for creating a dial-in password:

passwd -m ***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 superuser can change a dialup shell password.

Setting login restrictions

Login restrictions help protect your system by locking out unauthorized users. These restrictions make it difficult for an unauthorized user to use repeated login attempts to guess a user password and gain entry:

- Setting login restrictions on accounts (this page)
- Setting login restrictions on terminals (page 30)
- Locking or unlocking a user account (page 31)
- Locking or unlocking a terminal (page 32)

Setting login restrictions on accounts

You can prevent unauthorized use of an account by restricting the number of consecutive failed logins. When this limit is exceeded, the account is locked.

In the **Account Manager**, select a user name, then select **Login Controls** from the **Users** menu.

Enter a value in the "Failed login attempts allowed before account is locked" field or click on the **default of** button to select the system default.

To lock or unlock an account, see "Locking or unlocking a user account" (page 31).

To change the system default value, use this command line:

usermod -D -x "{maxLoginAttempts *value***}"**

You can change the value for an individual user with the **usermod**(ADM) command by omiting the **-D** option and appending the user name to the above command.

NOTE Terminal restrictions provide greater control over the login process.

See also:

- **usermod**(ADM) manual page (for command line interface)
- "Setting login restrictions on terminals" (this page)

Setting login restrictions on terminals

You can prevent unauthorized access to your system with login restrictions on terminals. Repeated login attempts are typically associated with trying to crack an account password.

Use the **Terminal Manager** located in the *System* directory of the SCOadmin hierarchy. To set login restrictions on all terminals, select **Defaults**. To change the settings for an individual terminal, select **Examine**.

You can control these attributes:

Consecutive unsuccessful logins
sets the number of consecutive unsuccessful login attempts before the terminal is locked. A value of "INFINITE" or "infinite" disables this type of lock for the terminal.

Delay between attempts
sets the interval between login prompts. The message `Wait for login retry:` is displayed between login attempts.

Time to complete login
sets the maximum interval for a login attempt. When this period expires, the login is considered unsuccessful and the process is restarted (unless the number of unsuccessful attempts has been exceeded).

After five login attempts, the login session is a aborted and a new one is started after the "Delay between login attempts" elapses. The five-login limit cannot be changed. When a login session occurs over a modem connection, the consequence of aborting the session is that the modem disconnects.

NOTE Locks on the system console are ignored when the superuser logs in. This is to avoid a complete lock-out of all users everywhere. Because this special login is allowed, you should physically protect the system console.

See also:

- "Logging unsuccessful login attempts" (page 243)

Locking or unlocking a user account

In the **Account Manager**, select a user name, then select **Login Controls** from the **Users** menu.

To lock an account, click on the **Lock Account** button. When an account has been locked, the **Unlock Account** button appears instead.

If a lock has been applied to the account, it is listed under "Current Account Lock Status". An account can be locked because:

- the administrator has locked the account
- there were too many unsuccessful login attempts
- the account password has expired

You can also unlock accounts from the command line:

passwd -u ***username***

To lock an account, use the **-l** option.

See also:

- "Controlling password expiration" (page 23)
- "Setting login restrictions on terminals" (page 30)

Locking or unlocking a terminal

A terminal refers to the device file associated with the user login session (for example, */dev/tty01* or */dev/ttyp00*).

Use the **Terminal Manager** located in the *System* directory of the SCOadmin hierarchy. To lock and unlock a terminal, respectively, select **Lock** or **Unlock**. When the prompt appears for the terminal, enter the name, (for example: **tty01**). When a terminal is locked, this message is displayed when an attempt is made to log in:

```
Terminal is disabled -- see Authentication Administrator
```

Assigning user powers

The administrator has the option of granting (or restricting) special powers:

- Assigning subsystem authorizations (this page) — the ability to run administrative programs
- Changing system privileges (page 35) — to set privileges on user processes
- Allowing users to skip login messages (page 37)
- Allowing users to execute superuser commands (page 37) — to run utilities otherwise only executable by root
- Controlling the use of job scheduling commands (page 39) — to schedule jobs with **cron**(C), **at**(C), and **batch**(C)

The Low and Traditional security profiles (page 42) assign most of the capabilities described here by default and it should not be necessary to change them. With the High (C2) security profiles, few are assigned by default; most privileges are intended for users entrusted with system administration.

Assigning subsystem authorizations

Authorizations allow users to run certain system programs. Primary authorizations (page 33) are intended for users entrusted with system administration. Secondary authorizations grant more limited capabilities.

In the **Account Manager**, select the user name, then select **Authorizations** from the **Users** menu.

To change authorizations, deselect the **Use system default authorizations for this user account** button. This allows you to assign a set of authorizations specific to this account.

To add an authorization, select an entry the "Not authorized" column and click on the **Add** button.

To remove an authorization, select an entry in the "Authorized" column and click on the **Remove** button.

To change the authorizations assigned by default, use this command:

usermod -D -x "{subsystemAuths {*list*}}"

where ***list*** is one or more authorizations separated by spaces.

You can change the value for an individual user with the **usermod**(ADM) command by omiting the **-D** option and appending the user name to the above command.

Primary authorizations

Primary authorizations effectively divide superuser powers into subsystems, allowing you to assign only the capabilities you want the user to have. Use secondary authorizations (page 34) to assign more limited capabilities to normal users.

Users lacking the required authorization to run a SCOadmin manager will see the message `You are not authorized to run...`

WARNING The **auth** subsystem authorization should only be assigned to persons entrusted with account administration. Never assign **auth** by default because it permits users to make changes to any account, including *root*. The **backup**, **sysadmin**, and **passwd** authorizations can be similarly abused — do not assign them lightly.

Table 1-2 Primary authorizations

Authorization	SCOadmin Manager	Powers
mem	–	access to system data tables, listing all processes on the system
terminal	–	unrestricted use of the **write**(C) command
lp	**Printer Manager**	administer printers
backup	**Backup Manager**	perform backups
auth	**Account Manager** **Terminal Manager**	administer accounts and terminals: adding users, changing passwords, controlling logins
audit	**Audit Manager**	run system audits and generate reports
cron	**Cron Manager**	control use of **cron**(C), **at**(C), and **batch**(C) commands
root	–	use any command found in */tcb/files/rootcmds* — see "Allowing users to execute superuser commands" (page 37)
sysadmin	**Filesystem Manager**	alter mount configuration
passwd	-	manage system passwords using **passwd**(C)

NOTE Certain SCOadmin managers require more than one authorization. For example, to run the **Backup Manager** (**backup** authorization), you also need the **sysadmin** authorization (to mount filesystems).

Secondary authorizations

Secondary authorizations allow limited access by users to resources that would otherwise be tightly controlled (for example, without the **printqueue** authorization, users would only be able to see their own jobs when they use the **lpstat** command). They are useful when running the Improved or High security profiles to provide behavior that is more consistent with other UNIX systems.

Table 1-3 Secondary authorizations

Secondary authorization	Parent authorization	Powers
audittrail	audit	generate personal audit reports on one's own activities
backup_create	backup	create (but not restore) backups
restore	backup	restore (but not create) backups
queryspace	backup	use **df**(C) command to query disk space
printqueue	lp	view all jobs in queue using **lpstat**(C)
printerstat	lp	use printer enable/disable commands
su	auth	access the *root* (superuser) account and other accounts. Access still requires a password; see "Accessing other accounts with su(C)" (page 38) for more information.
shutdown	root	use the **Shutdown Manager** or **shutdown** in conjunction with the **asroot**(ADM) command as described in "Allowing users to execute superuser commands" (page 37).

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.)

Changing system privileges

System privileges (page 36) allow user processes to execute specific operating system services. For example, the ability to change ownership of a file is governed by the **chown** privilege. (The **chown** privilege allows the use of the **chown**(S) system call that enables **chown**(C) to work.)

In the **Account Manager**, select the user name, then select **Privileges** from the **Users** menu.

To change privileges assigned, deselect the **Use system default privileges for this user account** button. This allows you to assign a set of privileges specific to this account.

To add a privilege, select an entry the "Not allowed" column and click on the **Add** button.

To remove a privilege, select an entry in the "Allowed" column and click on the **Remove** button.

To change the privileges assigned by default, use this command:

usermod -D -x "{privs {*list*}}"

where ***list*** is one or more privileges separated by spaces.

You can change the value for an individual user with the **usermod**(ADM) command by omiting the **-D** option and appending the user name to the above command.

Table 1-4 System privileges

Privilege	Allows user processes to
configaudit	configure audit subsystem parameters
writeaudit	write audit records to the audit trail
execsuid	run set-UID programs
chmodsugid	to set set-UID and set-GID bit on files
chown	to change the owner of an object
suspendaudit	suspend operating system auditing of the process

NOTE Restricted **chown** is required for NIST FIPS 151-1 conformance. The **chown** privilege should not be assigned to users if you wish to conform to NIST FIPS 151-1 requirements. The kernel parameter **CHOWN_RES** also controls this behavior; see the *Performance Guide* for more information.

Under the Low and Traditional security profiles, most system privileges are assigned by default and should not require modification. Under the High security profile, **chmodsugid** is not assigned by default. Most users require only **execsuid** to perform routine tasks. If the user needs to create files with the SUID or SGID bits, they must have **chmodsugid**. To change ownership of a file ("give it away"), the **chown** privilege is required. If a user does not have this privilege, ownership of files can only be changed by *root*. The audit privileges (**configaudit**, **writeaudit**, and **suspendaudit**) should never be assigned to anyone other than the audit administrator. They are intended for use by a program designed to run as a trusted application.

See also:

- "Audit privileges" (page 263)
- "SUID/SGID bits and security" (page 235)

System privileges and authorizations

If you are operating with the High and Improved security profiles, you must assign certain system privileges when you assign subsystem authorizations. Although most of these are already assigned by default, they are listed in Table 1-5, "Subsystem privilege requirements" (page 37) in case you modify the defaults. One exception is the audit subsystem, which requires the

addition of the **configaudit** and **suspendaudit** privileges. These privileges should never be assigned by default, or to ordinary users.

NOTE Under the Low and Traditional security profiles, most privileges are already assigned by default.

Table 1-5 Subsystem privilege requirements

Subsystem authorization	Privilege required
audit	configaudit, execsuid, writeaudit
auth	execsuid, chown
backup	execsuid
lp	chown
cron	chmodsugid, chown, execsuid
sysadmin	chmodsugid, chown, execsuid

Allowing users to skip login messages

You can permit users to skip the messages displayed at login time. To do this, make certain this line appears in */etc/default/login*:

```
ALLOWHUSH=YES
```

Users can then create an empty file called *.hushlogin* in their home directory that suppresses login messages.

Allowing users to execute superuser commands

It is possible to assign users the capability of executing *root*-only commands without giving them complete *root* powers. This is done using the **asroot**(ADM) utility to create a new authorization associated with the command you wish to assign. You can then add this authorization to any user. You can also assign the **root** subsystem authorization to permit the use of all commands configured with **asroot**. For example, when the **shutdown** authorization is assigned assigned to a user, they can execute the **shutdown** command like this:

/tcb/bin/asroot shutdown

The procedure for setting up a superuser command for this usage is described in detail in the **asroot**(ADM) manual page.

NOTE When the system is configured with the High security profile, the **asroot** utility requires that the user's password be entered.

See also:

- "Assigning subsystem authorizations" (page 32)
- "Accessing other accounts with su(C)" (this page)

Accessing other accounts with su(C)

The **su**(C) utility (for "superuser") can be used to switch over to another account temporarily. It is primarily used to access the *root* account, when it is executed without an argument. Otherwise, it is used in this form:

su *username*

su prompts for the account password, and if it is correct, a Bourne shell is started under the other account. Transitions with **su** do not affect the login user ID (LUID), so login and audit records remain accurate.

If a dash (-) is included in the command (**su -**), the environment for that user is executed (including login shell, home directory, and so forth), making it essentially the same as logging in as that user. To exit the shell, enter **exit** or press ⟨Ctrl⟩**D** and you are returned to your own account.

Users can **su** to an pseudo-user account if they own it. To access the *root* account (or any other account they are *not* responsible for), however, the user must have the **su** authorization. Refer to "Assigning subsystem authorizations" (page 32) for more information.

NOTE The Low, Traditional, and Improved security profiles assign the **su** authorization by default. Users can **su** to any account if they know the password. Under the High security profile, the **su** authorization is not assigned.

See also:

- "Logging su(C) usage" (page 39)
- "Assigning subsystem authorizations" (page 32)
- "Allowing users to execute superuser commands" (page 37)

Logging su(C) usage

Use of the **su**(C) command is logged in the file */usr/adm/sulog* like this:

```
SU 07/08 22:32 + ttyp0 mavrac-root
```

The entry indicates the date, time, location, and name of the account using the command. The following information is logged if an entry for **SULOG** appears in */etc/default/su*:

```
SULOG=/usr/adm/sulog
```

See also:

- "Checking and clearing system log files" (page 91)

Controlling the use of job scheduling commands

You can control use of the **cron**(C), **at**(C), and **batch**(C) commands that schedule or delay the execution of processes. Use the **Cron Manager** located in the *System* directory of the SCOadmin hierarchy.

The **Cron Manager** performs these tasks:

- Changing the default permissions for job scheduling (this page)
- Changing the job scheduling permissions for a user (page 40)
- Using environment files for the at or batch commands (page 40)

NOTE In addition to granting permission to use these commands, you must make sure that the users have the *chmodsugid* privilege (page 35).

See also:

- "Scheduling your processes" in the *Operating System User's Guide*

Changing the default permissions for job scheduling

The system is initially configured to permit use of the job scheduling commands by *root, sys, adm* and *uucp* only, denying access to other users.

To change the system defaults, make these selections in the **Cron Manager** (the first is for **cron** and the second for **at** and **batch**):

Authorize ➪ Scheduled ➪ Default
Authorize ➪ Delayed ➪ Default

These three selections are displayed:

None	Execution is not permitted for any users.
Allow	Allow all users to execute the command.
Deny	Deny all users access to the command.

The current behavior is highlighted. Use the arrow keys to select the behavior desired, or enter the first letter. Remember that users can be allowed or denied on an individual basis as well. The settings for individuals take precedence over the system defaults.

Changing the job scheduling permissions for a user

To change individual user permissions, make these selections in the Cron Manager (the first is for **cron** and the second for **at** and **batch**):

Authorize ➪ Scheduled ➪ User
Authorize ➪ Delayed ➪ User

The cursor is placed on the "User:" field. Enter the name of the user or press ⟨F3⟩ for a list of possible users. When the user name is selected, these selections are displayed:

Allow — Allow this user to execute the command.

Deny — Deny this user access to the command.

Use the arrow keys to select the behavior required. This setting overrides the system default.

Using environment files for the at or batch commands

It is also possible to define the environment in which **at** and **batch** commands execute. To edit the **at** and **batch** prototype files respectively, use these selections in the **Cron Manager**:

Authorize ➪ Environment ➪ At
Authorize ➪ Environment ➪ Batch

NOTE Only *root* can make these selections.

These options edit the files */usr/lib/cron/.proto* (for **at**) or */usr/lib/cron/.proto.b* (for **batch**). These files are placed at the start of the shell script formed for all **at** and **batch** jobs. This script must conform to the usual */bin/sh* syntax and contain some variables particular to the prototype file. These variables are:

$d Current directory of the user at the time of submission.

$l **ulimit** for the user at the time of submission.

$m **umask** for the user at the time of submission.

$t Time (in seconds past 1 Jan 1970) that the script is run.

$< Replaced with the entire script that the user submits. Normally, this appears last in the file, after the prologue that you set up. If you decide to include information after this variable, the shell script may exit before reaching it.

Only the superuser can edit these files.

Example environment files

There are many uses for prototype files; two examples are shown below:

- Run jobs in a particular queue at a lower priority by inserting a **nice**(C) command:

```
nice -5 /bin/sh << 'END_OF_FILE'
$<
END_OF_FILE
```

- Specify that a queue should execute commands using an alternative shell:

```
/bin/csh << 'END_OF_FILE'
$<
END_OF_FILE
```

For most sites, the prototype files provided with the distribution should be sufficient.

Changing the system security profile

You were asked to choose a security profile (page 42) at installation time. It is possible to later select a different profile by using the **Security Profile Manager** located in the *System/Security* directory of the SCOadmin hierarchy.

Use the **Current security profile** button to change the profile and select **Save** from the **Security** menu to save the new profile. You may be asked to reboot your system before the change takes effect.

WARNING After using lower security profiles it is possible to select the Improved or High defaults, but this does not mean your system conforms to the requirements of a C2 system. By definition, a C2 system must adhere to the requirements from initial installation. This is because modifications made to the system while at the lower level potentially violate those associated with the higher level.

These profiles are available:

High — recommended for systems containing confidential information and accessed by many users. Passwords are strictly controlled and assigned to users; users cannot choose their own. User accounts cannot be removed or reactivated. All C2 features are engaged and account database corruption results in a lockout of all users until the administrator fixes the problem.

Improved — recommended for systems accessed by groups of users who can share information. Password expiration is more lenient and users can choose their own passwords. LUIDs are not enforced, and user accounts can be removed or reactivated as desired. Account database corruption results in system lockout.

Traditional — Provided for compatibility with other UNIX systems. Passwords do not expire and standard System V password checking is used. Passwords are not required. Database corruption is handled transparently.

Low — Recommended only for systems which are not publicly accessible and which have a small number of cooperating users. No C2 features are engaged and no password checking is done. The */etc/shadow* does not exist by default.

The High and Improved defaults are designed to meet the requirements set forth by the Department of Defense's *Trusted Computer System Evaluation Criteria* (also known as TCSEC or the *Orange Book*).

You can change the security profile from the command line using **relax**(ADM). For example, this command sets the Improved profile:

relax improved

The security profiles are merely a set of values that can be customized as desired. If `The security subsystem has been modified` appears on the screen, that means that you have made changes to individual security parameters. Customized values are overwritten when you select a new profile.

Security profiles

A security profile is a set of pre-configured values for parameters that control the security behavior of your system, such as how long passwords last, or what privileges are assigned to users. Once you choose a profile, you can switch to another profile, or change any one of the dozens of parameters on an individual basis.

Table 1-6 System security profiles

Security parameters	Security profiles Low	Traditional	Improved	High
Passwords				
Minimum days between changes	0	0	0	14
Expiration time (days)	infinite	infinite	42	42
Lifetime (days)	infinite	infinite	365	90
User can choose own	yes	yes	yes	no
User can run generator	yes	yes	yes	yes
Maximum generated length	8	8	10	10
Minimum length	1	3	5	8
Password triviality checks	none	System V	**goodpw** weak[1]	**goodpw** strong[2]
Password obviousness checks	-	no	no[1]	yes[2]
Password required to login	no	no	yes	yes
Single user password required	yes	yes	yes	yes
Logins				
Maximum unsuccessful attempts (account/terminal)	infinite	99	5/9	3/5
Delay between login attempts (secs) — terminal only	0	1	2	2
Time to complete login (secs) — terminal only	60	60	60	60
Authorizations				
Primary	backup, lp, mem, terminal	mem, terminal,	*none*	*none*
Secondary	audittrail, queryspace, shutdown, su	audittrail, printqueue, queryspace, su	audittrail, queryspace, printqueue, su	queryspace
Privileges				
	chmodsugid, chown, execsuid, suspendaudit	chmodsugid, chown, execsuid	chmodsugid, chown, execsuid	chown, execsuid
Default umask[3]	022	022	027	077
C2 Features				
LUID enforcement[4]	no	no	no	yes
STOPIO on devices[4]	no	no	no	no
SUID/SGID clear on write[4]	no	yes	yes	yes
Users can be deleted[5]	yes	yes	no	no
Database corruption[6]	recover	recover	lockout	lockout
Database precedence[7]	System V	System V	TCB	TCB

(Continued on next page)

Table 1-6 System security profiles
(Continued)

Other				
Users can schedule jobs	allow	allow	deny	deny
Home directory permissions	755	755	750	700
Dialup printers allowed	yes	yes	no	no
Hushlogin allowed[8]	yes	yes	yes	no
Password for asroot(ADM)	no	no	no	yes
Significant characters in passwords	8	8	80	80
su(C) use logged	no	yes	yes	yes
/etc/shadow present	no	yes	yes	yes

Notes:

1. Simple checks are made, such as ensuring at least three characters differ and that at least one character be non-alphabetic.
2. Thorough checks are made, including disallowing words that appear in the online dictionary.
3. 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.
4. These features are explained in "Disabling C2 features" (page 254).
5. A requirement central to C2 is that a user ID (UID) cannot be reused. This means that user accounts cannot be reused or reactivated after retirement. With the lower security profiles, user accounts can be removed rather than retired and user IDs can be altered or reused.
6. On a system that conforms to C2 requirements, users are locked out of a system when a security database becomes corrupted. This ensures that the system does not operate in a potentially non-secure state. In the lower defaults, the system attempts to correct inconsistencies automatically and displays a warning rather than locking out users.
7. Two sets of account databases are maintained: UNIX System V and trusted computing base (TCB) files. One set is used as a master when a discrepancy occurs. This is described in "Configuring database precedence and recovery" (page 46).
8. This feature allows the suppression of login messages. See the **login**(M) manual page for information.

Understanding account database files

An important distinction between UNIX systems is how account information is stored. This affects the interaction of accounts across different types of UNIX systems, and governs how programs access this data. The account database files fall into two categories: UNIX system files (those defined in the *System V Interface Definition*) and the Trusted Computing Base (TCB) files that extend System V security. These files are supported and maintained by the system to ensure compatibility with other UNIX systems.

System V files:

- */etc/passwd*. This publicly readable file is present on most UNIX systems and contains both account data (user ID number, login shell, and so forth) and (on some systems) an encrypted account password. Password aging information is also supported. The format is documented in *passwd*(F). It can be edited by experienced administrators, but using the **Account Manager** is the preferred method for adding and maintaining user accounts — see "Editing the /etc/passwd file" (page 46).
- */etc/shadow*. This file is readable only by *root*. It contains the encrypted password otherwise found in the */etc/passwd* file. The format is documented in the *shadow*(F) manual page. This file exists by default in all security profiles except Low, where it still can be created using **pwconv**(ADM). See "Configuring the shadow password file" (page 47).
- */etc/default/passwd* and */etc/default/login*. These contain default account information and are documented in **passwd**(C) and **login**(M), respectively. (In many cases, information in these files is duplicated in the Protected Password and System Defaults database.)

TCB files:

- *Protected Password* database (*/tcb/files/auth/[a-z]/username*). This database implements the requirements for the C2 level of trust as defined by the *Trusted Computing System Evaluation Criteria* (TCSEC). It contains the encrypted password of the user. If the user has specific system privileges, password parameters, and so forth that override the System Defaults database, they are stored here. The format of this file is described in *authcap*(F).
- *System Defaults* database (*/etc/auth/system/default*). This contains the system-wide account defaults. The contents of this file are determined by the security profiles selected (Low, Traditional, Improved, or High). The contents of this file can be changed dynamically to affect all user accounts, unless a user has specific values set. The format of this file is described in the *authcap*(F) manual page.

All database files are updated automatically when a change is made from the **Account Manager** or the command line.

NOTE In the event of a discrepancy between these files, either the UNIX System V files or the TCB databases are used as the master to bring them into agreement. In the Low and Traditional security profiles (page 42) the UNIX System V files are the master. You can also configure which set of files is used as the master set — see "Configuring database precedence and recovery" (page 46).

Configuring database precedence and recovery

When the Low or Traditional security profiles are configured on your system, inconsistencies between the TCB and UNIX System V database files are handled transparently without interrupting system operation. Under the higher security profiles, the TCB database files take precedence and any corruption or inconsistencies that occur result in a lockout of non-*root* users until the problem is corrected.

This behavior can be set independently of the security profile with the **usermod**(ADM) command.

To reconfigure database precedence, use this command:

usermod -D -x "{tcbDatabaseIsMaster *value*}"

where ***value*** is either 1 (yes) or 0 (no). If you set ***value*** to 0, the UNIX System V database files described in "Understanding account database files" (page 44) are used as the master. The non-master database files are maintained only for consistency and are not relied upon for data used by the system.

To reconfigure how the system treats inconsistencies, use this command:

usermod -D -x "{integrityRequired *value*}"

where ***value*** is either 1 (lock out all users until problem is fixed) or 0 (generate warnings but do not lock out users). If set to 1, the administrator must log in on the override terminal as described in "Using the override terminal" (page 248).

See also:

- "Dealing with filesystem and database corruption" (page 246)
- "Changing the system security profile" (page 41)
- "Password compatibility across UNIX systems" (page 49)

Editing the /etc/passwd file

Although using the **Account Manager** is preferred for adding and maintaining users and groups, administrators accustomed to editing */etc/passwd* directly can simply do so — unless your system is configured with the Improved or High security profiles. If this is so, you must also run the **authck**(ADM) command with the **-p** and **-s** options to update the accounts database. With the lower security profiles, you can edit */etc/passwd* without running **authck**.

See also:

- "Understanding account database files" (page 44)
- "Configuring database precedence and recovery" (page 46)
- "Configuring the shadow password file" (this page)
- **useradd**(ADM) manual page (for command line interface)

Configuring the shadow password file

The */etc/shadow* file contains the encrypted passwords otherwise found in */etc/passwd*. If your system does not have the shadow file configured, you can create it with the **pwconv**(ADM) command. **pwconv** creates a new *shadow* file and removes the encrypted passwords from */etc/passwd*. (When the */etc/shadow* exists, */etc/passwd* contains an "x" in the password field.) Use **pwunconv**(ADM) to remove the *shadow* file (and incorporate passwords into */etc/passwd*).

NOTE The */etc/shadow* exists by default under all security profiles except Low.

See also:

- *shadow*(F) manual page
- *passwd*(F) manual page

Copying user accounts

Use the **ap**(ADM) utility to copy user accounts from other SCO systems. **ap** creates a profile containing all account data for one or more users. You need not use **ap** if NIS is configured.

WARNING The **ap** utility does not create profiles that are portable to non-SCO UNIX systems. If you need to migrate UNIX accounts from a different vendor or from an SCO® XENIX® system, use the **addxusers**(ADM) utility described in "Copying user accounts from SCO XENIX or non-SCO UNIX systems" (page 48).

Account information is gathered from the */etc/passwd* file and the Protected Password database. Irrelevant information about the user (including unsuccessful logins, unsuccessful password changes, and the location and time of last login) is not included in the profile.

To create a profile and install it on another machine:

1. Log in as *root* and enter this command on the machine where the accounts reside:

   ap -d -v *usernames* > profile.acct

   where ***usernames*** is the list of one or more user names.

2. Log in as *root* and move the *profile.acct* file to the target machine (use **tar**, or **cp** if your machine is on a network).

3. Enter this command:

   ap -r -f profile.acct *usernames*

   The new account information is in place and ready for use.

See also:

- "Administering NIS users and groups" in the *Networking Guide*
- "Copying user accounts to non-SCO UNIX systems" (this page)
- "Copying user accounts from SCO XENIX or non-SCO UNIX systems" (this page)

Copying user accounts to non-SCO UNIX systems

You can migrate user accounts to other XENIX or UNIX systems by copying the */etc/passwd* (and */etc/shadow*, if applicable) to the target system. If the target system does not use */etc/shadow*, run the **pwunconv** utility to consolidate the information into */etc/passwd* before copying the file to the target system.

NOTE If you need to migrate accounts to another SCO system, use the **ap**(ADM) utility described in "Copying user accounts" (page 47).

See also:

- "Copying user accounts from SCO XENIX or non-SCO UNIX systems" (this page)

Copying user accounts from SCO XENIX or non-SCO UNIX systems

Use the **addxusers**(ADM) utility to copy accounts from non-SCO UNIX systems (or SCO XENIX systems) to your system. **addxusers** accepts an edited */etc/passwd* file as input and makes the necessary database modifications for use on your system. Refer to the **addxusers**(ADM) manual page for a complete procedure.

See also:

- "Copying user accounts" (page 47)
- "Password compatibility across UNIX systems" (this page)

Password compatibility across UNIX systems

The password encryption scheme used by your SCO system maintains compatibility with other XENIX and UNIX system implementations, while providing the ability to create passwords with more than eight significant characters. However, if you are using the Improved or High security profiles, passwords of up to 80 characters are allowed, and these cannot be imported to UNIX systems from other vendors. To ensure compatibility, you can limit the number of characters considered significant when passwords are checked and encrypted.

To reconfigure the number of significant segments, use the command:

usermod -D -x "{passwdSignificantSegments *value*}"

where ***value*** is from 1 to 10. A value of 1 (instead of the 10 used in the Improved or High defaults) allows the encryption mechanism to ignore characters following the first eight, thus allowing complete compatibility with other systems.

WARNING Most XENIX and UNIX systems will accept passwords longer than eight characters, but only use the first eight for encryption. This can cause unexpected results when moving an encrypted password string from one of these systems to an SCO system. If a password longer than eight characters has been used, such as "narcissus", only the first eight characters ("narcissu") should be entered on an SCO system.

See also:

- "Understanding account database files" (page 44)
- "Changing the system security profile" (page 41)

Troubleshooting the Account Manager

Most error messages displayed by the **Account Manager** are self-explanatory. In other cases, the error message will be generic, for example:

```
Unable to create new user account: name
```

Error boxes of this type include a **Details** button to provide you with additional information. The problems reported fall into these categories:

- Illegal specification for a user or group attribute (this page)
- Remote administration problem (this page)
- Missing or corrupted database files (this page)

Illegal specification for a user or group attribute

When creating or modifying users and groups, these conditions will cause an error:

- a colon (:) or newline character in any attribute (including user name, home directory, comment)
- a misspelled or nonexistent login shell (or path name)
- an illegal user or group name (names cannot begin with a number or exceed 8 characters in length)
- an illegal UID or GID number (they must fall within the minimum and maximum values listed in the **User** or **Group** selections of the **Options** menu in the **Account Manager**)

Remote administration problem

If you are performing remote administration, the remote system may be unreachable or there may be a configuration problem. You should also make certain there is user equivalence on the remote machine. See "Troubleshooting TCP/IP" in the *Networking Guide* or "Adding user equivalence" in the *Networking Guide* for more information.

Missing or corrupted database files

The **Details** may indicate file data (or the file itself) is missing, for example:

```
The user name does not exist in /etc/passwd
```

This means that an account database entry in */etc/passwd* is corrupted or the file itself is missing. Table 1-7 (page 51) contains a list of critical files accessed by the **Account Manager**. Follow these steps to solve the problem:

1. Use the **tcbck**(ADM) command to determine if any files are missing:

   tcbck

   This will list any missing database files.

2. If no files are reported missing, or if only the file */tcb/files/auth/r/root* is missing, use the **authck**(ADM) command to check the security databases and resolve inconsistencies:

   authck -a -y

   authck will regenerate the Protected Password database entry for root (*/tcb/files/auth/r/root*) and any other users it finds missing.

3. Check any files reported as missing in step 1 with **more**(C) to see if they contain information. If the command reports a file as not found or the file appears to be empty, follow the procedure in "Restoring critical security database files" in the *SCO OpenServer Handbook*. If there appears to be no problem, try the **Account Manager** again.

Table 1-7 Security-related files

Filename	Purpose
/etc/auth/system/default	Default control database
/etc/auth/system/authorize	Authorization database
/etc/default/accounts	Account/group creation defaults
/etc/group	Group database
/etc/passwd	Account database
/tcb/files/auth	Protected Password database directory

See also:

- "Troubleshooting SCOadmin" in the *SCO OpenServer Handbook*
- "Checking the security databases" in the *SCO OpenServer Handbook*
- "Dealing with filesystem and database corruption" (page 246)
- "Checking system configuration" in the *SCO OpenServer Handbook*

Chapter 2

Administering filesystems

One of the most important responsibilities of a system administrator is creating and maintaining filesystems. Filesystem maintenance includes keeping the filesystems clean, repairing damaged filesystems, and ensuring adequate space for all users. Filesystem administration includes:

- Creating filesystems on the primary hard disk (at installation time) — see "The installation and upgrade procedure" in the *SCO OpenServer Handbook*
- Adding filesystems to the primary or secondary hard disk after installation — see "Partitioning a hard disk using fdisk" and "Dividing a disk partition into divisions using divvy" in the *SCO OpenServer Handbook*
- Allowing filesystems to be mounted across the network — see "Exporting and unexporting filesystems" in the *Networking Guide*
- Mounting and unmounting filesystems (page 74)
- Checking and repairing filesystems (page 79)
- Adding and removing mount configuration (page 58)
- Maintaining free space in filesystems (page 85)
- Maintaining filesystem efficiency (page 97)

See also:

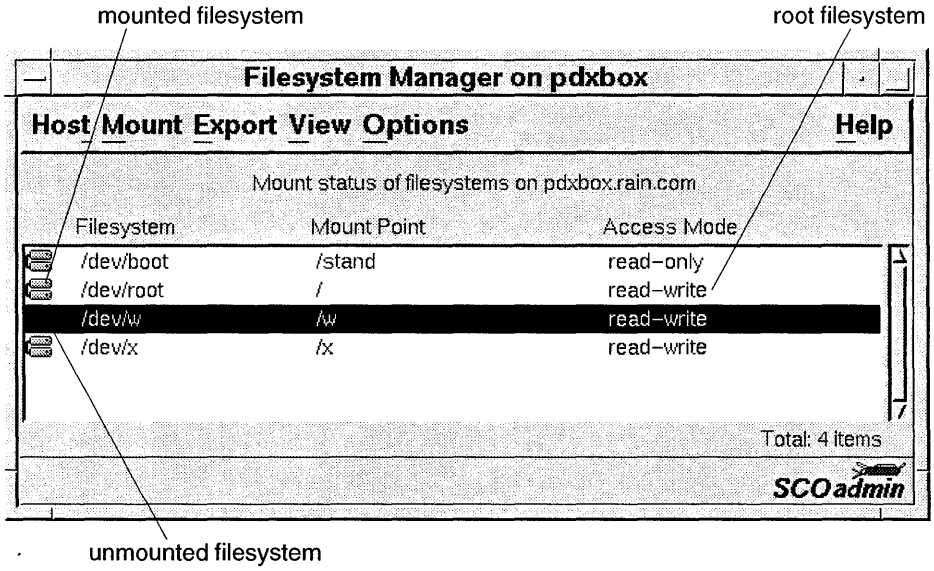
- "About filesystems" (page 55)
- "The Filesystem Manager interface" (page 54)
- "Creating filesystems on floppy disks" (page 77)
- "Adding support for different filesystem types" (page 57)

The Filesystem Manager interface

Use the **Filesystem Manager** to administer the filesystems on your system. You can start the **Filesystem Manager** in any of these ways:

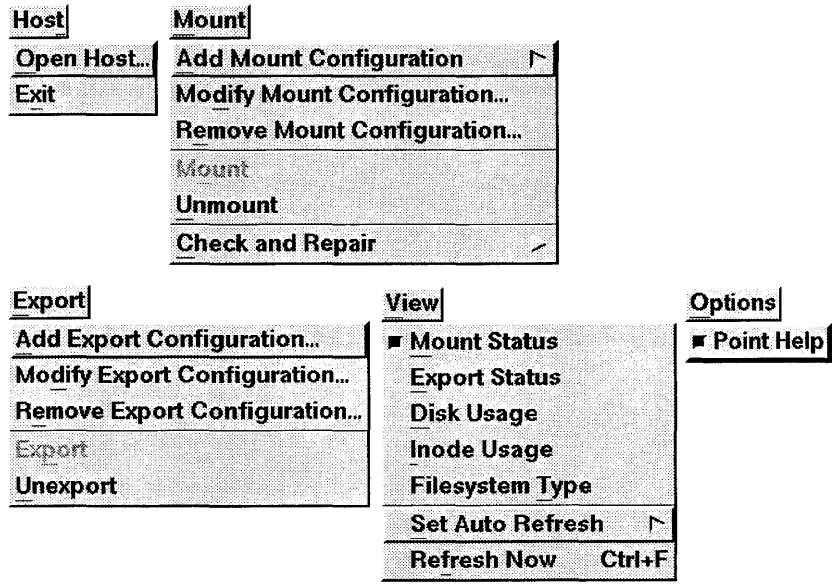
- Double-click on the **Filesystem Manager** icon in the *Filesystems* directory of the *System Administration* window on the Desktop.
- Start the SCOadmin launcher by entering **scoadmin** on the command line, then selecting **Filesystems**, then selecting **Filesystem Manager**.
- Enter **scoadmin filesystem manager** on the command line (or abbreviate to **scoadmin fi**).

For more information on using SCOadmin managers, see "Administering your system with SCOadmin" in the *SCO OpenServer Handbook*.



Authorization

When run from an account other than *root*, use of the **Filesystem Manager** requires the **sysadmin** authorization. See "Assigning subsystem authorizations" (page 32) for more information.



About filesystems

A "filesystem" is a distinct division of the operating system, consisting of files, directories, and the information needed to locate and access them.

Filesystems can reside on local hard disks, CD-ROMs, and floppy disks. You can also mount remote filesystems on your local system and make local filesystems available for other systems to mount. See "Exporting and unexporting filesystems" and "About mounting and unmounting NFS filesystems" in the *Networking Guide*.

Each UNIX system has at least two filesystems on the primary hard disk. The main filesystem is called the *"root"* filesystem (also known by the symbol " / "). The *root* filesystem contains the programs and directories that comprise the operating system. On small hard disks, the *root* filesystem also includes all the user directories. The second filesystem is called */stand* and contains the information needed to boot the system, the **boot** program and the kernel, */stand/unix*.

The primary hard disk can be divided into more than these two filesystems. See "The installation and upgrade procedure" in the *SCO OpenServer Handbook* for information about dividing the hard disk at installation time. Dividing the primary hard disk into multiple filesystems protects the data and makes maintenance easier. The most common division is the */u* or */home* filesystem,

used for user accounts. Keeping user accounts separate from the *root* filesystem makes backing up your system much easier. Because system data changes less frequently than user data, you can back up the */u* or */home* filesystem more frequently than the */* (*root*) filesystem. See Chapter 3, "Backing up filesystems" (page 105) for more information.

The system administrator creates the filesystems on the hard disk, then mounts and unmounts (page 74) — attaches and detaches — the filesystem when needed, similar to the way you access a floppy disk.

See also:

- "Filesystem types" (this page)

Filesystem types

In the **Filesystem Manager** (page 54), view the types of filesystem on your system by selecting **Filesystem Type** from the **View** menu.

Table 2-1 Supported filesystem types

Type	Filesystem
HTFS (page 64)	High Throughput Filesystem (default)
EAFS (page 64)	Extended Acer Fast Filesystem
AFS (page 64)	Acer Fast Filesystem
S51K (page 64)	AT&T UNIX System V 1KB Filesystem
DTFS (page 67)	Compression Filesystem
HS (page 70)	High Sierra CD-ROM Filesystem
ISO9660 (page 70)	ISO 9600 CD-ROM Filesystem
Rockridge (page 72)	Rockridge CD-ROM Filesystem
XENIX	XENIX Filesystem
DOS (page 73)	DOS Filesystem
NFS	Network Filesystem
NetWare — see "Filesystem mount options (NUC)" in the *Guide to Gateways for LAN Servers*	SCO Gateway for NetWare Filesystem

In addition, your SCO system supports LMCFS (LAN Manager Client Filesystems), although these cannot be managed with the **Filesystem Manager**.

See also:

- "Distributed filesystems" in the *Networking Guide*
- **fstyp**(ADM) manual page
- **mkdev**(ADM) manual page
- **mkfs**(ADM) manual page

Adding support for different filesystem types

Depending on which SCO system you have, some of the filesystem device drivers might not be configured into the kernel by default. If the filesystem driver is not already configured, you must add it to the system using the **Hardware/Kernel Manager** before the system recognizes the filesystem type. (The **Hardware/Kernel Manager** runs the **mkdev**(ADM) script in */usr/lib/mkdev*).

The **ht** driver, which provides support for the HTFS (default), EAFS, AFS, and S51K filesystem types, is always configured into the default kernel.

Table 2-2 Filesystem drivers

Filesystem	mkdev script
DOS Filesystem	dos
DTFS Filesystem	dtfs
HTFS Filesystem	htfs
High Sierra/ISO9660/Rockridge Filesystem	high-sierra
XENIX Filesystem	xenix

To add support for a filesystem type, configure the driver into the kernel using the **Hardware/Kernel Manager**. See "Configuring drivers with the Hardware/Kernel Manager" in the *SCO OpenServer Handbook*.

After adding the driver to the system configuration, create and relink a new kernel and then reboot the system for the change to take effect. See "Relinking the kernel" in the *SCO OpenServer Handbook*.

NOTE You cannot use the **Hardware/Kernel Manager** to configure support for the NFS, SCO Gateway for NetWare, and LMCFS filesystems. For these filesystems, see "Configuring protocols" in *Configuring Network Connections*.

Adding and removing mount configuration

Once a filesystem has been created (on the hard disk, floppy disk, or CD-ROM) or exported from a remote server, you must add the mount information to the system so that you can then mount and use the filesystem.

To add filesystem mount information, in the **Filesystem Manager** (page 54):

1. Select **Add Mount Configuration** from the **Mount** menu, then select **Local** or **Remote**.
2. Select the appropriate filesystem:

   Local Select the filesystem device file from the pop-up list on the local system. If the device file does not appear in the list, enter the correct device in the "Device File" field.

   Remote Select the type of remote filesystem, select the server from the list of hosts known to the local system, then select the directory or volume.
3. Specify the filesystem parameters. See "Modifying filesystem mount configuration" (page 59).

This adds the filesystem mount configuration to the */etc/default/filesys* file and adds the filesystem to the **Filesystem Manager** list.

NOTE Remote administration (using the **Open Host** selection of the **Host** menu) requires user equivalence on the machine you plan to administer. The remote machine must recognize the account being used to administer users on the local machine. See "Adding user equivalence" in the *Networking Guide* for more information. As on the local machine, non-*root* accounts require the **sysadmin** authorization to run the **Filesystem Manager**. See "Assigning subsystem authorizations" (page 32) for more information.

If you selected "Now" to mount the filesystem immediately, the **Mount** icon appears to the left of the filesystem name indicating that the filesystem is mounted.

To remove the mount configuration for a filesystem, select **Remove Mount Configuration** from the **Mount** menu and click on **Yes** at the prompt.

This removes the mount information from the */etc/default/filesys* file and the filesystem no longer appears in the **Filesystem Manager** list. If the filesystem is mounted, removing the mount configuration also unmounts the filesystem.

Modifying filesystem mount configuration

In the **Filesystem Manager** (page 54), select a filesystem from the list, select **Modify Mount Configuration** from the **Mount** menu, then change the filesystem parameters:

Mount Point specifies the directory where you want to attach ("mount") the filesystem. For example, the *u* filesystem is generally mounted on the */u* directory.

Description specifies the (optional) description of the filesystem. For example, the default description for the *root* filesystem is "The root filesystem".

Filesystem Type specifies the type of filesystem. See Table 2-1, "Supported filesystem types" (page 56).

The filesystem type is set when the filesystem is created and cannot be changed.

Access mode

Read-only

mounts the filesystem as read-only so that no changes can be made to the filesystem while it is mounted. You must mount CD-ROM filesystems and filesystems on write-protected floppy disks as read-only to prevent errors.

Read-write

mounts the filesystem with write permissions enabled. Changes can be made to the filesystem while it is mounted.

NOTE When mounting remote filesystems, read/write permissions might already be limited by the NFS server. For more information, see "Setting export access permissions" in the *Networking Guide.*

Can Users Mount specifies that regular users can mount and unmount the filesystem. Unless you select this option, only *root* can mount and unmount the filesystem. See "Enabling users to mount filesystems" (page 63).

When to Mount

Now

mounts the filesystem immediately; the filesystem remains mounted until you unmount it or you reboot the system.

At System Startup

does not mount the filesystem immediately. The next time you boot the system, the filesystem is mounted automatically.

Check and Repair Options

changes the filesystem checking mount options. See "Check and repair options" (page 80). These options are not available for CD-ROM filesystems (High Sierra, ISO9660, Rockridge), NFS-mounted filesystems, or DOS filesystems; you cannot check and repair these filesystem types with the **Filesystem Manager** or **fsck**(ADM).

Advanced Options changes the filesystem-specific advanced options. Which advanced options are available depends on the type of filesystem you are modifying (these options are not available for XENIX filesystems). See:

- "Filesystem mount options (HTFS, EAFS, AFS, S51K)" (page 64)
- "Filesystem mount options (DTFS)" (page 67)
- "Filesystem mount options (High Sierra and ISO9660)" (page 70)

- "Filesystem mount options (Rockridge)" (page 72)
- "Filesystem mount options (DOS)" (page 73)
- "NFS filesystem advanced mount options" in the *Networking Guide*
- "Filesystem mount options (NUC)" in the *Guide to Gateways for LAN Servers*

See also:

- *filesys*(F) manual page

Modifying HTFS, EAFS, AFS, and S51K root filesystem mount configuration

You cannot modify the mount configuration for an HTFS, EAFS, AFS, or S51K *root* filesystem from the **Filesystem Manager**. Modifying the *root* filesystem mount options requires relinking the kernel.

To modify the configuration for these filesystems:

1. Log in as *root* on the system where you want to change the *root* filesystem mount configuration.
2. Use the **Hardware/Kernel Manager** as described in "Configuration tools" in the *Performance Guide*. Select category 10, "Filesystem configuration".
   - To enable logging (page 66), set **ROOTLOG** to 1 (default).
   - To disable logging, set **ROOTLOG** to 0.
   - To enable checkpointing (page 65), set **ROOTCHKPT** to 1 (default).
   - To disable checkpointing, set **ROOTCHKPT** to 0.
   - To enable versioning (page 67) and set the maximum number (*n*) of versioned files, set **ROOTMAXVDEPTH** to a non-zero number.
   - To disable versioning, set **ROOTMAXVDEPTH** to 0 (default).
   - To set the number of seconds (*n*) before a file is versioned, set **ROOTMINVTIME** to *n* (a non-zero number). The default value of **ROOTMINVTIME** is 0.
3. Relink the kernel with the new *root* filesystem parameters — see "Relinking the kernel" in the *SCO OpenServer Handbook*.
4. Reboot the system by entering:

   reboot

The mount configuration you specified is applied to the HTFS, EAFS, AFS, or

S51K *root* filesystem.

See also:

- "Filesystem mount options (HTFS, EAFS, AFS, S51K)" (page 64)

Modifying DTFS root filesystem mount configuration

You cannot modify the mount configuration for a DTFS *root* filesystem from the **Filesystem Manager**.

To modify the configuration for these filesystems:

1. Log in as *root* on the system where you want to change the *root* filesystem mount configuration.
2. Use the **Hardware/Kernel Manager** as described in "Configuration tools" in the *Performance Guide*. Select category 10, "Filesystem configuration."
   - To enable sync on close (page 70), set **ROOTSYNC** to 1.
   - To disable sync on close, set **ROOTSYNC** to 0 (default).
   - To enable compression (page 68), set **ROOTNOCOMP** to 0 (default).
   - To disable compression, set **ROOTNOCOMP** to 1.
   - To enable versioning (page 67) and set the maximum number (*n*) of versioned files, set **ROOTMAXVDEPTH** to a non-zero number.
   - To disable versioning, set **ROOTMAXVDEPTH** to 0 (default).
   - To set the number of seconds (*n*) before a file is versioned, set **ROOTMINVTIME** to *n* (a non-zero number). The default value of **ROOTMINVTIME** is 0.
3. Relink the kernel with the new *root* filesystem parameters — see "Relinking the kernel" in the *SCO OpenServer Handbook*.
4. Reboot the system by entering:

   reboot

The mount configuration you specified is applied to the DTFS *root* filesystem.

See also:

- "Filesystem mount options (DTFS)" (page 67)

Enabling users to mount filesystems

In the **Filesystem Manager** (page 54), select **Add Mount Configuration** or **Modify Mount Configuration** from the **Mount** menu, then select **Yes** for "Can Users Mount".

NOTE You can also enable user mounting transparently using **automount**(NADM). If you create an **automount** map for a filesystem exported from a remote machine, the filesystem is mounted automatically whenever the filesystem or one of its files is accessed. Such a mount is transparent to the user, and the filesystem unmounts automatically after a certain period of inactivity. See Chapter 13, "Configuring the NFS automounter" in the *Networking Guide* for details.

Under normal circumstances, only *root*, programs run as *root*, and users with the **sysadmin** and **backup** (or **queryspace**) authorizations can mount and unmount filesystems with the **Filesystem Manager** or the **mount**(ADM) and **umount**(ADM) commands. However, when you select this mount option for a filesystem, you can enable users to mount and unmount the filesystem using the **mnt**(C) and **umnt**(C) commands. The **mnt** command allows users other than *root* to access the functionality of **mount** for that particular filesystem.

For example, if you enable users, they can mount the */usr* filesystem by entering:

mnt /usr

To unmount the filesystem, users enter:

umnt /usr

NOTE For reasons of security, it is not a good idea to allow users to mount floppy filesystems.

Instruct your users in these proper mounting procedures:

- Unmount filesystems when not in regular use.
- You cannot use **mnt** and **umnt** if your current working directory is under the mount point for the filesystem that you want to unmount. For example, if the current working directory is */usr/mark* and you try to unmount */usr*, you see:

  `unmount: filesystem busy: Device busy`

  To unmount the filesystem, you must change to a directory that is not under the filesystem mount point and issue the **umnt** command again.
- You can display a list of mounted filesystems with **mnt**. The **mnt -t** command lists all the local and remote filesystems known to the system (the contents of the */etc/default/filesys* file). If the value of "Can Users Mount" is **No**, regular users cannot mount the filesystem.

See also:

- "Enabling user mounting of SCO Gateway for NetWare filesystems" in the *Guide to Gateways for LAN Servers*
- *filesys*(F) manual page
- **su**(C) manual page
- **asroot**(ADM) manual page
- **chmod**(C) manual page

Filesystem mount options (HTFS, EAFS, AFS, S51K)

The **Filesystem Manager** supports the following mount options for non-*root* HTFS, EAFS, AFS, and S51K filesystems; for *root* filesystems, see "Modifying HTFS, EAFS, AFS, and S51K root filesystem mount configuration" (page 61).

Mount as Temporary Filesystem
: mounts the filesystem as a temporary data area (such as */tmp*). This improves system performance because the system updates the information less frequently. (The potential for loss of data is increased as a result.)

Checkpointing
: transitions the filesystem to a clean (consistent) state at regular intervals. This reduces the probability that your filesystem will need cleaning if the system is halted unexpectedly.

Logging
: performs "intent logging", recording filesystem transactions to a log file before they are committed to disk. This increases data availability by reducing the checking and repairing the filesystem to a few seconds. This time is independent of filesystem size.

These options apply to HTFS filesystems only:

Maximum number of file versions
: determines maximum number of undeletable (versioned) files allowed on the filesystem. A value of 0 disables versioning.

Minimum time before a file is versioned
: sets minimum time before a file is versioned. If set to 0, a file is always versioned (as long as **Maxiumum number of file versions** is greater than 0). If set to a value greater than 0, the file is versioned after it has existed for that number of seconds.

See also:

- "Mounting as a temporary filesystem" (this page)
- "Checkpointing your filesystem" (this page)
- "Logging filesystem transactions" (page 66)
- "Versioning filesystems (undelete)" (page 67)

Mounting as a temporary filesystem

To improve performance, your temporary filesystem (for example */tmp*, */u/tmp*, or */usr/tmp*), can be set up as either an EAFS, AFS, S51K or HTFS filesystem type. If you have any temporary filesystems, you can select this option so they are mounted as temporary filesystems automatically every time the system is rebooted.

Temporary filesystems are updated less frequently and are recommended for use on filesystems containing temporary data only. If this option is used on */tmp*, the overall system performance may improve.

CAUTION Some applications use */tmp* to save data. If this option is enabled, then the "checkpointing" feature will be disabled automatically.

Checkpointing your filesystem

Checkpointing is the process of transitioning filesystems to a clean (consistent) state. Filesystem data consists of user file data (the contents of a file) and the data structures used to store the data (also known as "meta data"). Recently-accessed data is held in memory ("cached") for a short time in case it is needed again. If the system is stopped unexpectedly, this cached data can be lost.

By default, HTFS, EAFS, AFS, and S51K filesystems periodically "checkpoint" (write) cached meta data back to disk. This increases the probability that the filesystem meta data will be in a consistent state if the system is halted unexpectedly. (There may be a small loss of user data, which is not checkpointed.)

If your system should experience a system error, checkpointing will reduce the likelihood that the filesystem will need to be checked and repaired with **fsck**(ADM) when rebooting, thus mimimizing downtime.

NOTE While checkpointing is appropriate for most users of HTFS, there is a small performance penalty. To achieve maximum throughput, you should consider disabling checkpointing.

See also:

- **fsck**(ADM) manual page

Logging filesystem transactions

Intent logging minimizes system downtime after abnormal shutdowns by logging filesystem transactions. When the system is halted unexpectedly, this log can be replayed and outstanding transactions completed. The check and repair time for filesystems can be reduced to a few seconds, regardless of the filesystem size.

NOTE Intent logging does not increase the reliability of filesystem. Only transactions concerning file meta data (the structures concerned with *storing* data) are logged. The purpose is to minimize system downtime.

The ability to locate and check only affected areas of the disk for inconsistencies is central to the logging mechanism. The structure of the mechanism is:

- A log file is created as special file in the root of each mounted filesystem. (This file is normally invisible and is neither readable nor removable when logging is enabled. When logging is disabled on a previously enabled filesystem, the log file appears as *.ilog0000*.)
- The location of the file is kept in the superblock (an area at the top of the filesystem structure which describes the attributes of the filesystem in terms of type, size, available space and latest modification time).
- Changes to a file's meta data (as opposed to the data it contains) such as inodes and block bitmaps are cached in memory. Before making any changes on disk, a log entry is written synchronously. After the changes are performed, a "transaction complete" indication is stored on disk.

If the system crashes before the log is written, it is as if the change (any modifications to the filesystem) never occurred. If the system crashes after the log is written, but before the transaction complete, **fsck** either completes the change or undoes it. If the system crashes after the transaction is completed, then the modification has been completed, and there is nothing for **fsck** to do.

NOTE For optimum benefit, both logging and checkpointing should be enabled. Checkpointing marks the filesystem as clean whenever the filesystem is inactive. If the system needs to be checked (in the event of an abnormal system halt, for instance) only areas of the disk manipulated since the last checkpoint operation need to be examined for inconsistencies.

See also:

- **fsck**(ADM) manual page

Versioning filesystems (undelete)

Versioning allows deleted files to be recovered with the **undelete**(C) command or on the Desktop as described in Chapter 12, "Deleting and recovering files and directories" in *Using the Desktop*. If versioning is enabled on a filesystem, files and directories can be designated for versioning and administered as described in "Retrieving deleted files" in the *Operating System User's Guide*. The versioning feature can be enabled system-wide or for individual filesystems.

To enable versioning on all non-root DTFS/HTFS filesystems:

1. Use the **Hardware/Kernel Manager** as described in "Configuration tools" in the *Performance Guide*. Select category 10, "Filesystem configuration."
   - To enable versioning (this page) and set the maximum number (*n*) of versioned files, set **MAXVDEPTH** to a non-zero number.
   - To disable versioning, set **MAXVDEPTH** to 0 (default).
   - To set the number of seconds (*n*) before a file is versioned, set **MINVTIME** to *n* (a non-zero number). The default value of **MINVTIME** is 0.
2. Relink the kernel with the new filesystem parameters — see "Relinking the kernel" in the *SCO OpenServer Handbook*.
3. Reboot the system by entering:

   reboot

To enable versioning on a per-filesystem basis, see "Filesystem mount options (HTFS, EAFS, AFS, S51K)" (page 64) and "Filesystem mount options (DTFS)" (this page).

To enable versioning on the root filesystem, see "Modifying HTFS, EAFS, AFS, and S51K root filesystem mount configuration" (page 61) or "Modifying HTFS, EAFS, AFS, and S51K root filesystem mount configuration" (page 61).

Filesystem mount options (DTFS)

The **Filesystem Manager** supports the following mount options for non-*root* DTFS filesystems; for *root* filesystems, see "Modifying DTFS root filesystem mount configuration" (page 62).

Compression compresses the data on your hard disk, thereby increasing the storage capacity on your disk. The compression ratio depends on the type of file. This feature is enabled by default.

Sync-on-Close writes files to disk as soon as they are closed. This ensures data integrity and robustness because you can switch the machine off immediately afterward without loss of data.

Maximum number of file versions
sets maximum number of undeletable (versioned) files allowed on the filesystem. A value of 0 disables versioning.

Minimum time before a file is versioned
sets minimum time before a file is versioned. If set to 0, a file is always versioned (as long as **Maxiumum number of file versions** is greater than 0). If set to a value greater than 0, the file is versioned after it has existed for that number of seconds.

WARNING The sync-on-close feature can significantly degrade system performance because of the increased writing to disk.

See also:

- "Data compression" (this page)
- "Forced data writes to disk" (page 70)
- "Versioning filesystems (undelete)" (page 67)

Data compression

The Compression Filesystem (DTFS) uses transparent data compression and a more efficient disk media format to increase the storage capacity of your disks. The compression is performed prior to the writing of file-data blocks to disk. It is designed primarily for use on systems where disk space is limited, such as personal workstations. The compression ratios are dependent on the data type.

Table 2-3 Average disk savings

File type	Average savings (%)
Directories	34
Executable programs	36
Program source files	40
Archives and dynamically loaded libraries	42
Symbolic links	50
Binary data (bitmaps, word processing, databases, spreadsheets, and so on)	55
ASCII data (log files, configuration files, and so on)	60

Standard utilities may be used to perform backup and restore operations on this filesystem.

On other filesystems, the number of disk blocks reported by **ls -s** does not include the amount of space consumed by the disk inode structure. With DTFS, the size of the disk inode is included in the number of blocks reported. This might make a small file appear as if it occupies more space on a DTFS filesystem when compared to other filesystems.

When displaying disk usage information, you can display:

- the uncompressed size of files (in bytes) with the **ls -l** command
- the actual number of physical 512-byte blocks used by the file with the **ls -s** command

For example, entering:

```
cd /usr/adm
ls -ls messages
```

might show:

```
82 -rw-r--r--   1 bin    bin    106295 Apr 08 23:01 messages
```

indicating that the *messages* file, whose logical size is 106,295 bytes, consumes only 82 disk blocks, or a total of 41,984 bytes of disk space. This represents a 60% savings.

The **du**, **df**, and **quot** utilities display compressed sizes in blocks.

Forced data writes to disk

Sync-on-close ensures that all files modified by a process are written back to the disk when they are closed. This minimizes data loss in the event of power failure.

In addition, the filesystem is transitioned to a quiescent state every second. DTFS does not depend on the standard filesystem flusher functionality (**bdflush**) to synchronize user data; it does this itself.

In addition, a feature called "shadow paging" ensures that new file data is first written to shadow blocks, leaving the original blocks unmodified. If a system failure occurs before the new data is written to disk, the original data is still intact because the blocks were not freed.

These features mean that as soon as you have saved the file, you can switch off the power and the file will be on the disk. This is particularly useful in desktop and laptop environments.

WARNING Some caching disk controllers use write-back caching techniques that nullify the improved data reliability and integrity provided by DTFS. If this is the case, data integrity cannot be guaranteed. However, controllers that use write-through caching take advantage of DTFS features.

If your controller supports both caching techniques, configure it to use write-through caching.

If sync-on-close is enabled, it will noticeably degrade your overall system performance because of the time spent writing to the disk.

Filesystem mount options (High Sierra and ISO9660)

When you add or modify a High Sierra or ISO9660 CD-ROM filesystem, you can set advanced options for this filesystem type. The **Filesystem Manager** supports the following options:

Convert Filenames to Lowercase displays filenames in lowercase and suppresses any trailing period (.) characters. The default is to leave filenames in uppercase. If filenames have been recorded as POSIX® filenames, this option is not available.

Show Hidden Files displays any hidden files when you mount the filesystem. The default is to not display hidden files.

Show File Version Numbers displays file version numbers when you mount the filesystem. The default is to not display version numbers. If filenames have been recorded as POSIX filenames, this option is not available.

Default User defines the default user for files and directories that are recorded without a final unrestricted extended attribute record (XAR). XAR is the optional data structure used within an ISO9660 format CD-ROM for recording file attributes.

If you set this option, you must use a valid user name or a valid user ID between 0 and 60000. If you do not set this option, the files are recorded with the user ID of the mount point for the filesystem.

Default Group defines the default group for files and directories that are recorded without a final unrestricted XAR.

If you set this option, you must use a valid group name or a valid group ID between 0 and 60000. If you do not set this option, the files are recorded with the group ID of the mount point for the filesystem.

Default File Permissions sets the permissions for files recorded without a final unrestricted XAR. Specify the mode in octal. See the **chmod**(C) manual page. The default permissions are set to 0555.

Default Directory Permissions sets the permissions for directories recorded without a final unrestricted XAR. Specify the mode in octal. See the **chmod**(C) manual page. The default permissions are set to 0555.

See also:

- "Mounting and unmounting filesystems" (page 74)
- **mount**(ADM) manual page

Filesystem mount options (Rockridge)

When you add or modify a Rockridge CD-ROM filesystem, you can set advanced options for this filesystem type. The **Filesystem Manager** supports the following options:

SUID and SGID file permissions
specifies whether to honor or ignore **setuid** and **setgid** bits on a file when executing it. See "SUID/SGID bits and security" (page 235) and the **setuid**(S), and **setgid**(S) manual pages for more information.

Device files and named pipes writable
specifies that device files and named pipes on the mounted CD-ROM filesystem are writable.

If you do not make device files and named pipes writable, opening a file on a read-only filesystem displays an EROFS error. However, if you do make device files and named pipes writable, the mounted filesystem is not XPG3 compliant.

User ID Mapping maps the user ID recorded on the CD-ROM to a user ID on the system.

If you set this option, you must use a valid user name or a valid user ID between 0 and 60000.

You can use user ID mappings already defined in a file. Specify mappings in the file using one of the following formats:

```
cd_uid:UID comment
cd_uid:user_name comment
```

where **cd_uid** is the user ID as recorded on the CD-ROM. Put each user ID mapping on a separate line.

Group ID Mapping maps the group ID recorded on the CD-ROM to a group ID on the system.

If you set this option, you must use a valid group name or a valid group ID between 0 and 60000.

You can use group ID mappings already defined in a file. Specify mappings in the file using one of the following formats:

```
cd_gid:GID comment
cd_gid:group_name comment
```

where **cd_gid** is the group ID as recorded on the CD-ROM. Put each group ID mapping on a separate line.

Device ID Mapping maps the device number recorded on the CD-ROM to a device number on the system.

If you specify this option, the device pathname must be a full pathname from the root of the CD-ROM. The device number on the system must be a valid device number in the range of 0 to 65535.

You can use device mappings already defined in a file. Specify mappings in the file using the following format:

`cd_path:`*device_#* *comment*

where **cd_path** is the full pathname from the root of the CD-ROM filesystem of the device node to map. Put each device mapping on a separate line.

NOTE The **Filesystem Manager** supports one only of each of the user ID, group ID, and device ID mapping options for Rockridge filesystems. If you require more than one user, group, or device mapping, define the mappings in a file (using the formats listed above) and specify that file when modifying these options with the **Filesystem Manager**.

See also:

- "Mounting and unmounting filesystems" (page 74)
- **mount**(ADM) manual page (for command-line interface)
- "Imported filesystems" (page 240)
- "SUID, SGID, and sticky bit clearing on writes" (page 236)

Filesystem mount options (DOS)

Filenames in DOS are uppercase. When you mount a DOS filesystem, you can choose to convert the DOS filenames to lowercase. The default behavior when mounting DOS filesystems is to leave DOS filenames uppercase.

See also:

- "About mounting DOS filesystems" (page 75)
- "Mounting and unmounting filesystems" (page 74)
- **mount**(ADM) manual page (for command-line interface)
- **doscmd**(C) manual page

Mounting and unmounting filesystems

In the **Filesystem Manager** (page 54), select the filesystem to mount from the list, then select **Mount** from the **Mount** menu.

If the filesystem is not listed, you must add the mount configuration for the filesystem before you can mount it. See "Adding and removing mount configuration" (page 58).

If an attempt to mount a filesystem fails, check the filesystem before mounting the filesystem again. See "Checking and repairing filesystems" (page 79).

To unmount a filesystem, select the filesystem, then select **Unmount** from the **Mount** menu. At the confirmation prompt, click on **Yes**.

If your current working directory is under the mount point for the filesystem that you are unmounting, the unmount action fails with the following error:

```
unmount: filesystem busy:  Device busy
```

You must change to a directory that is not under the filesystem mount point and select **Unmount** from the **Mount** menu again.

To view the mount status of the filesystems in the list, select **Mount Status** from the **View** menu.

NOTE Remote administration (using the **Open Host** selection of the **Host** menu) requires user equivalence on the machine you plan to administer. The remote machine must recognize the account being used to administer users on the local machine. See "Adding user equivalence" in the *Networking Guide* for more information. As on the local machine, non-*root* accounts require the **sysadmin** authorization to run the **Filesystem Manager**. See "Assigning subsystem authorizations" (page 32) for more information.

See also:

- "Modifying filesystem mount configuration" (page 59)
- "About mounting DOS filesystems" (page 75)
- "About mounting and unmounting NFS filesystems" in the *Networking Guide*
- "Mounting and unmounting SCO Gateway for NetWare filesystems" in the *Guide to Gateways for LAN Servers*
- **mount**(ADM) manual page (for command-line interface)
- **umount**(ADM) manual page (for command-line interface)
- *filesys*(F) manual page

About mounting DOS filesystems

In addition to manipulating DOS files with the DOS utilities provided with the operating system, you can mount a DOS filesystem and access its files freely while still operating from your UNIX system.

NOTE The ability to mount DOS filesystems is available, regardless of whether you have SCO® Merge™ installed.

When you mount a DOS filesystem, you can edit, examine, or copy DOS data and text files, without first copying them into the UNIX filesystem. However, you cannot execute DOS files and applications from a mounted DOS filesystem. To do this, you must run SCO Merge or boot DOS from a DOS partition.

If you use the DOS utilities on a mounted DOS filesystem, you see this error:

```
dosdir: FAT not recognized on /dev/dsk/0sC
```

In addition, you cannot create DOS filesystems using the **mkfs**(ADM) command. The DOS mounting feature is intended for existing DOS filesystems (on a floppy disk or on an existing DOS partition).

The UNIX operating system handles mounted DOS filesystems, without actually changing the files, by superimposing certain qualities of UNIX system filesystems on the DOS filesystem. UNIX filesystems are highly structured and operate in a multiuser environment. Thus, many UNIX filesystem concepts do not apply to DOS, including:

- file ownership
- access permissions
- special files (pipes, device files, and so forth)
- links

To make DOS files readily accessible, the UNIX system superimposes access permissions and file ownership on the DOS filesystem when you mount it. See "DOS filesystems and access permissions" (page 76).

Because no changes are made to the DOS files, the carriage return character (^M) is visible when you edit a DOS file on a UNIX system. (UNIX systems use only a newline character; DOS uses both a carriage return and a newline.) To switch the end-of-line from DOS format to UNIX system format, use **dtox**(C). To switch back to UNIX system format, use **xtod**(C). See "Displaying a DOS file" and "Converting DOS files to and from UNIX system file format" in the *Operating System User's Guide.*

See also:

- "DOS filesystem limitations" (page 77)
- Chapter 7, "Using other operating systems with an SCO system" in the *SCO OpenServer Handbook*
- Chapter 6, "Working with DOS" in the *Operating System User's Guide*
- **doscmd**(C) manual page
- **tr**(C) manual page

DOS filesystems and access permissions

Only *root* and users with **filesystem** authorization can mount filesystems, including DOS filesystems.

A user's access to a mounted DOS filesystem is determined by the permissions and ownership that *root* places on the filesystem.

When DOS files are mounted on a UNIX system:

- The permissions and ownership of the filesystem are determined by the permissions of the mount point.

  For example, if *root* creates a mount point */x* with permissions of **0777**, all users can read or write the contents of the filesystem. If the mount point is owned by *root*, all files in the DOS filesystem and any created by other users are owned by *root*.

- The permissions for regular files on the mounted DOS filesystem are set to either **0777** (for readable/writable files) or **0555** for read-only files.
- The permissions are based on the **umask**(C) of the creator when a user creates a file on a mounted DOS filesystem.

  For example, if a user's **umask** is **0022**, all files created by that user have permissions of **0644**.

- Files in a DOS filesystem cannot have multiple links.

  NOTE "." and ".." are special cases; on a mounted DOS filesystem, they are not links as they are on a UNIX system.

- Features such as locking govern how different users can access a file simultaneously on UNIX systems. 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.

DOS filesystem limitations

The following limitations apply when accessing files on a mounted DOS filesystem:

- The rules for filenames and filename conversion follow the guidelines listed in the **doscmd**(C) manual page. In addition, the standard DOS restrictions on illegal characters apply. However, you can use wildcards to match DOS files, just as you do on a UNIX system.
- Timestamps differ between DOS and UNIX systems.

  When mounted from the UNIX system partition, the UNIX DOS filesystem driver records the creation, modification, and access times of files in GMT (Greenwich Mean Time) and then converts them to the appropriate local time. DOS timestamps are recorded in the local time.

  Thus, in time zones other than GMT, when you access a file created in the DOS filesystem while mounted on the UNIX system, the timestamp will be wrong because DOS assumes that the timestamp was recorded in the local time. If you mount the DOS filesystem under the UNIX system and access a file created under DOS, the timestamp will also be wrong because the UNIX system assumes the timestamp was recorded in GMT and converts it to the local time.
- You cannot use the **Backup Manager** or the **backup**(ADM) utility to make backups of a mounted DOS filesystem. However, you can use DOS utilities and other copy programs such as **tar**(C) to create backups.

See also:

- "About mounting DOS filesystems" (page 75)
- **doscmd**(C) manual page

Creating filesystems on floppy disks

Creating a filesystem on a floppy disk is similar to creating a filesystem on a hard disk. Floppy disk filesystems are portable and can be mounted on any UNIX system. Use the special directory, */mnt*, to mount filesystems that do not have a specified mounting point.

To make a portable filesystem on a floppy disk, use the **Floppy Filesystem Manager** located in the *Filesystems* directory of the SCOadmin hierarchy:

1. Enter the number of the floppy disk type on which you are creating the filesystem.

   For example, enter **4** to create a filesystem on a 1.44 MB 3.5 inch disk (135tpi, double-sided, 18-sectors-per-track).

2. If you have more than one floppy disk drive, you are prompted for the disk drive (0 or 1) on which to create the floppy disk filesystem. Enter the drive number and press ⟨Enter⟩.

   For example, to create the filesystem on the primary floppy disk drive, enter **0**.

3. When prompted, insert the floppy disk in the appropriate drive and press ⟨Enter⟩.

4. If you have already formatted the disk, enter **n** at the prompt; the filesystem is created immediately.
   If you have not formatted the disk, enter **y**. You see messages like:

   ```
   formatting /dev/type
   track 00 head 0
   ```

   The track and head numbers count up as the disk is formatted. (If the */etc/default/format* file contains **VERIFY=Y**, the format is also verified after formatting.) See the **format**(C) manual page.

5. At the prompt for filesystem type, enter **y** to use EAFS, the default filesystem type. To use a different filesystem type, enter **n** and enter the filesystem type at the prompt. Valid filesystem types for floppy disk filesystems are AFS, DTFS, EAFS, HTFS, S51K, and XENIX. See Table 2-1, "Supported filesystem types" (page 56).

6. You see messages as the filesystem is created and then checked with **fsck**(ADM). Press ⟨Enter⟩ to return to the **Floppy Filesystem** menu. Enter **q** to quit.

Your floppy disk now contains a filesystem. Mount the filesystem directly from the command line, using **mount**(ADM) or add the filesystem mount information to the system and mount the filesystem with the **Filesystem Manager**. Once you mount a floppy disk filesystem, you can use it just as you would a hard disk filesystem.

NOTE For system security, do not allow users to mount floppy disk filesystems. However, if your system requires that users mount floppy disk filesystems, you must first add the mount configuration for the filesystem to the system and select the "Can Users Mount" option. See "Enabling users to mount filesystems" (page 63).

See also:

- "Adding and removing mount configuration" (page 58)
- "Mounting and unmounting filesystems" (page 74)
- **mkdev**(ADM) manual page

- **mount**(ADM) manual page (for command line interface)

Checking and repairing filesystems

When you boot the system after an abnormal shutdown, the **fsck**(ADM) utility runs automatically on the *root* filesystem. You can also set up your other filesystems to be checked automatically before they are mounted — See "Modifying filesystem mount configuration" (page 59).

NOTE You cannot check the *root* filesystem with the **Filesystem Manager**. To check the *root* filesystem, you put the system in single-user mode and run **fsck**(ADM).

To check filesystems manually, in the **Filesystem Manager** (page 54):

1. Select the filesystem to check from the list.
2. Select **Check and Repair** from the **Mount** menu, then select **Selected Filesystem**. If the filesystem is not in the list, select **Other Local Filesystem**.

   NOTE You can only check local filesystems. Check NFS filesystems on the system on which they reside. You can do this remotely using the **Open Host** selection of the **Host** menu. This requires user equivalence on the machine you plan to administer, as described in "Adding user equivalence" in the *Networking Guide*. As on the local machine, non-*root* accounts require the **sysadmin** authorization to run the **Filesystem Manager**. See "Assigning subsystem authorizations" (page 32) for more information.

3. Specify the type of checking and what to do if the filesystem is corrupted, then click on **OK**. See "Check and repair options" (page 80).

The **Filesystem Manager** checks the filesystem and displays the output in a Status window; see "Filesystem check phases (HTFS, EAFS, AFS, S51K)" (page 81). When the filesystem has been checked, click on **Close**.

NOTE In cases where a file appears to be lost, check for the file in the *lost+found* directory at the top of the filesystem. Files that have become disconnected from the structure are stored there. If the file is not in *lost+found*, restore the file from your backups. See "Restoring files from a scheduled filesystem backup" (page 131).

DTFS filesystems do not require *lost+found* directories.

See also:

- "How UNIX systems maintain files and filesystems" (page 83)
- **fsck**(ADM) manual page (for command-line interface)

Check and repair options

The following check and repair options are available for all filesystem types (except NFS, SCO Gateway for NetWare, DOS, HS, ISO9660, and Rockridge filesystems):

Check Filesystem Before Mounting

- Always
- Never
- Only if Dirty

What to do if Corrupted

- Automatic repair

  This is equivalent to responding "yes" to **fsck** prompts.

- Do not repair

  This is equivalent to responding "no" to **fsck** prompts.

> **NOTE** When you use the **Filesystem Manager**, the filesystem check and repair process is non-interactive. To check and repair your filesystems interactively, use **fsck**(ADM) from the command line.

Type of Checking

- Full Check

  Performs all five filesystem check phases. See "Filesystem check phases (HTFS, EAFS, AFS, S51K)" (page 81).

- Fast Check

  Performs Phase 1 (Check Blocks and Sizes) and Phase 5 (Check Free List Bitmap) only.

Create lost+found directory if none exists

Creates the *lost+found* directory, where **fsck** places unreferenced files. See filesystem check Phase 3 (page 82).

This option is not available for XENIX filesystems.

See also:

- "Modifying filesystem mount configuration" (page 59)
- "Checking and repairing filesystems" (page 79)
- **fsck**(ADM) manual page (for command line interface)

Filesystem check phases (HTFS, EAFS, AFS, S51K)

When you check and repair a filesystem, the **fsck**(ADM) utility scans and examines the filesystem structures, reporting on its progress with these messages:

```
** Phase 0 - Replay Log
** Phase 1 - Check Blocks and Sizes
** Phase 1b - Rescan For More DUPS
** Phase 2 - Check Pathnames
** Phase 3 - Check Connectivity
** Phase 4 - Check Reference Counts
** Phase 5 - Check Free List Bitmap
** Phase 6 - Salvage Free List Bitmap
```

NOTE DTFS filesystem check phases are different. See "Filesystem check phases (DTFS)" (page 82) for more information.

Each phase compares components and checks that these components agree with each other.

Phase 0 If intent logging (page 66) is enabled on the filesystem and a full check is not requested, a "fast check" is performed. This completes outstanding transactions found in the filesystem log and marks the filesystem clean. The remaining phases are skipped in this instance.

Phase 1 In Phase 1, **fsck** reads the inode table to determine sizes and locates the blocks used by each file. Inodes are checked for inode type, zero link counts, inode sizes, and bad or duplicate blocks. (Bad blocks are block values outside the boundaries of a filesystem; a duplicate block means that two inodes point to the same block on the disk.) When **fsck** clears an inode, it removes the bad information in the inode, which removes the file or directory that is associated with it. **fsck** also confirms the filesystem fits on the associated device. **fsck** attempts to locate the original and duplicate inodes for correction in Phase 2.

Phase 1b If duplicate blocks were found, the filesystem is scanned once again.

Phase 2 In Phase 2, **fsck** cleans up error conditions caused by improper inode status, out-of-range inode pointers, and directories that point to bad inodes. Files removed in Phase 1 must also have their directory entries removed. If files with duplicate blocks are found in

Phase 1, **fsck** removes both files.

Phase 3 In Phase 3, **fsck** checks for directory connectivity and reconnects files that were severed from the directory structure. Any files that are unreferenced but still valid are placed in the special *lost+found* directory in the *root* directory of the filesystem. For *root*, this directory is */lost+found*. When the directory is severed, the name of the file is lost, so **fsck** assigns a new name, the inode number of the file. See about inodes (page 84) for a description of inodes.

> **NOTE** **fsck** does not create or extend the *lost+found* directory. There must already be a sufficient number of empty slots in the directory for use by **fsck** when reconnecting files. When you add filesystem mount configuration with the **Filesystem Manager**, you can direct it to create a *lost+found* directory if none exists. See "Check and repair options" (page 80).

Phase 4 In Phase 4, **fsck** checks the link count of each entry that survived Phases 2 and 3. In some cases, files that were not referenced under the directory structure, but still have an inode, can be relinked to the filesystem in */lost+found*. Inodes that cannot be recovered are cleared.

Phase 5 In Phase 5, **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** rebuilds the free-block list.

Phase 6 If specified in Phase 5, **fsck**(ADM) reconstructs a free-block list from the altered filesystem during Phase 6.

For a complete list of error messages, see the **fsck**(ADM) manual page.

See also:

- "Checking and repairing filesystems" (page 79)
- "Check and repair options" (page 80)

Filesystem check phases (DTFS)

When you check and repair a DTFS filesystem, **fsck** works non-interactively to repair inconsistencies, reporting on its progress with these messages:

```
** Phase 1 - Check Super Block
** Phase 2 - Check File Structure
** Phase 3 - Check Directory Entries
** Phase 4 - Check Block Bitmap
```

Each phase compares components and checks that these components agree

with each other.

Phase 1 **fsck** reads the map of filesystem inodes (*inode bitmap*) and initializes the map of filesystem blocks (*block bitmap*).

Phase 2 **fsck** reads the inodes and validates them. DTFS files are managed through a structure known as a "B+ tree". The tree consists of disk block addresses that point either to intermediate blocks (known as "interior nodes") or to the actual user data blocks (referred to as "leaf nodes"). This tree structure is normally balanced (the distance from top level to leaf is the same for all paths in the tree). **fsck** verifies the parent inodes and rebalances any inodes (trees) that have become unbalanced.

Phase 3 **fsck** rebuilds the directory hierarchy. Because DTFS inodes contain the inode number of the parent directory and the inode filename, reconstruction is much easier and requires no *lost+found* directory. If any inodes that cannot be reconnected or contain invalid data (bad inodes) are found, the inode bitmap is updated. The block bitmap is then updated.

Phase 4 **fsck** updates information in the superblock.

For a complete list of error messages, see the **fsck**(ADM) manual page.

See also:

- "Checking and repairing filesystems" (page 79)
- "How UNIX systems maintain files and filesystems" (this page)

How UNIX systems maintain files and filesystems

Filesystem data is not stored on the hard disk in locations that correspond to individual files. On the contrary, the data is probably scattered across the disk. The data is spread around because the operating system does not really deal with files, but rather with units of data. For example, when you create a file, it might be stored on one part of the disk. If you edit that file and delete a few sentences here and there, you now use less disk space than you did before. This space amounts to a series of gaps in the area where your file was stored. Because disk space is a precious commodity, the system allocates those small amounts of disk space to other files.

Each filesystem contains special structures that allow the operating system to access and maintain the files and data stored on the filesystem:

Data blocks A "block" is a 1024-byte unit of data stored on the disk. (DTFS filesystems use variable block sizes to maximize use of space.) A data block can contain either directory entries or file data. A

directory entry consists of an inode number, a filename, and a version number for **undelete**(C) (file versioning).

Inodes An "inode" (information node) contains all the information about a file (except file data), including its location, size, file type, permissions, owner, and the number of directory entries linked to the file. The inode also contains the locations of all the data that make up a file so the operating system can collect it all when needed. The only information the inode does not contain is the name of the file and the contents; directories contain the actual filenames. In DTFS filesystems, inodes contain the inode number of its parent directory and the inode's filename. In addition, inodes are not statically allocated at filesystem creation as with other filesystem types. The number of free inodes in DTFS filesystems varies depending on the amount of free space available.

Superblock One special data block, the "superblock", contains overall information about the filesystem, just as the inode contains information about a specific file. The superblock contains the information necessary to mount a filesystem and access its data, including the size of the filesystem, the number of free inodes, and information about free space available. When the filesystem is mounted, the system reads information from the disk version of the superblock into memory.

Buffers To minimize seeking data on the hard disk, recently used data blocks are held in a cache of special memory structures called "buffers". Buffers make the operating system more efficient. Depending on the filesystem type and the setting of kernel parameters, the buffer cache is "flushed" (written to the disk) at set intervals.

Several configurable filesystem mechanisms affect how transactions are managed and committed. Some involve tradeoffs in performance against data integrity, others tradeoff performance against system recovery time.

Intent logging (page 66): When this feature is enabled, filesystem transactions are recorded in a log and then committed to disk. This increases system recovery speed with a very small performance penalty.

Checkpointing (page 65): When enabled, each filesystem is marked clean at regular intervals. If a filesystem is clean when the system halts, it will not be necessary for it to be checked by **fsck**. Like intent logging (which works in tandem with checkpointing), there is a small performance penalty.

Sync-on-close (DTFS) (page 70): When enabled, file data is immediately written to disk when a file is closed, imitating DOS behavior. This feature significantly degrades system performance.

See also:

- "Filesystem check phases (HTFS, EAFS, AFS, S51K)" (page 81)
- Chapter 5, "Tuning I/O resources" in the *Performance Guide*

Maintaining free space in filesystems

One important responsibility of the system administrator is ensuring that there is adequate disk space for the users. To do this, you must monitor the free space in each filesystem and take corrective action whenever free space gets too low. The amount of free space depends on the size of the disk containing the filesystem and the number of files on the disk. To determine how much free space is available in a filesystem, see "Displaying filesystem and directory usage statistics" (page 86).

A UNIX system operates best when at least 15% of the space in each filesystem is free. If a filesystem has less than 15% free space, system operation typically becomes sluggish.

When a filesystem has little or no space left to work, the system displays the following:

```
NOTICE: Out of space on EAFS dev hd (major/minor)
```

When the filesystem runs out of space, the system stops any attempts to write to the filesystem. (The error message list the filesystem by its major and minor device numbers, for example *hd(1/42)* for the root filesystem.) The only way to restore system operation is to delete or reduce files from the named filesystem.

If the free space falls below 15%:

1. Remove files from the */tmp* directory.

   By default, **cron** clears out the */tmp* and */usr/tmp* directories once each day. You can define which directories and how often they are cleared in the */etc/default/cleantmp* file. See the **cleantmp**(ADM) manual page.

2. Remind users to remove unused files.

   You can include a reminder in the */etc/motd* (message of the day) file, send e-mail, or send a message to the terminals of all users currently logged in. See the **mail**(C) and **wall**(ADM) ("write to all") manual pages.

3. Locate large files and ask owners to remove them.

   See Finding files of a certain size (page 88).

4. Locate system files to remove.

   See "Finding temporary files" (page 89).

5. Clear system log files.

   See "Checking and clearing system log files" (page 91).

6. Back up and remove unused data.

   For example, delete data in the */usr/adm*, such as the **sar** data in the */usr/adm/sa* directory or, if accounting is enabled, the data in the */usr/adm/acct* directory.

   Back up these files using the **Backup Manager** before removing them. See "Running unscheduled filesystem backups" (page 119).

7. Compress infrequently used files.

   See the **compress**(C) manual page.

8. Archive files into one larger file to free up inodes. See the **tar**(C) manual page.

9. Add more disk space.

   See "Adding disk space and restructuring filesystems" (page 94).

10. Add another disk and increase the size of the filesystem using the **Virtual Disk Manager**.

    See "Creating filesystems on virtual disks" (page 386).

11. If you are unable to add more disk space and your filesystem is not DTFS, you can regain some space by recreating your filesystems and changing the filesystem type to DTFS. DTFS compression filesystems (page 67) allow you to compress the data to increase storage capacity. See "Adding disk space and restructuring filesystems" (page 94) for instructions on revising your disk layout. When recreating the filesystem, remember to change the filesystem type to DTFS.

Displaying filesystem and directory usage statistics

In the **Filesystem Manager** (page 54), display disk usage statistics by selecting **Disk Space** from the **View** menu. The display changes to show (in MB):

- the total amount of disk space in the filesystem
- the amount of space that is currently being used
- the amount of free disk space

- the percentage of the filesystem that is currently being used

To display inode usage statistics, select **Inode Usage** from the **View** menu. The display changes to show:

- the total number of inodes currently being used
- the number of free inodes
- the percentage of the inodes that is currently being used

NOTE The view mode you select is saved between sessions with the **Filesystem Manager**.

To display the number of blocks used in a directory, use **du**(C) from the UNIX command line:

du *directory*

The optional *directory* must be the name of a directory in a mounted filesystem. If you do not give a directory name, **du** displays the number of blocks in the current directory. (**du** reports the value in 512-byte blocks by default; to display the value in 1024-byte blocks, include the **-k** option.)

For example, to display the number of blocks used in the directory */usr/james*, enter:

du /usr/james

The command displays the name of each directory in the */usr/james* directory and the number of blocks used.

Use the **-a** option to display each file in the specified directory; **-s** displays the grand total for the directory. For example, enter the following for the total number of blocks in */usr/james*:

du -s /usr/james

You see:

```
49790   /usr/james
```

To display a list of users and the number of blocks they own, use **quot**(ADM) from the UNIX command line:

quot *filesystem*

For example, to display the owners of files in */dev/usr*, enter:

quot /dev/usr

The command displays the users who have files in the filesystem and the numbers of blocks in these files:

```
/dev/usr:
74534   james
49262   johnd
36506   root
15470   bin
```

Locating files

You can locate all files with a specified name, permissions setting, size, type, owner, or last access or modification date using **find**(C). Use this command to locate seldom-used or excessively large files, files owned by a particular user, temporary files, *core* files, and any unused *a.out* files.

The syntax of **find** is:

find ***pathname option***

pathname is the pathname of the directory to search. The **find** command searches recursively, downward through all the directories under the named directory, for files that match the criteria specified by ***option***.

NOTE You must include the **-print** option for **find** to display the list of files that match the search criteria.

Finding specific files by name (-name)

For example, to locate and display all files named *temp* recursively in the */usr* directory, enter:

find /usr -name temp -print

Finding files of a certain size (-size)

For example, to locate and print a list of all the files greater than three blocks in size in all the directories (/ and below), enter:

find / -size +3 -print

Finding files by owner (-user)

For example, to find all files in the */work* directory owned by *olivier*, enter:

find /work -user olivier -print

Finding files of a certain type (-type)

For example, to locate all the directories in */usr/spool/uucp*, enter:

find /usr/spool/uucp -type d -print

Finding files by permissions (-perm *onum*)

onum is the octal number used with **chmod**(C).

For example, to locate and display all the files in the */usr* directory that give all users read, write, and execute permissions (***onum*** is

0777), enter:

find /usr -perm 0777 -print

See also:

- "Finding temporary files" (this page)
- "Executing commands based on find output" (this page)

Finding temporary files

Use **find** with the **-name** option to locate temporary files for removal.

A temporary file contains data created as an intermediate step during execution of a program. This file may be left behind if a program contained an error or was prematurely stopped by the user. The name of a temporary file depends on the program that created it. In most cases, the user has no use for temporary files, and you can safely remove them.

Use **find** to locate files with a specific name, such as *temp*. For example, to locate and display all files named *temp* under the */usr* directory, enter:

find /usr -name temp -print

When searching for temporary files, it is a good idea to search for files that have not been accessed for a reasonable period of time. For example, to find all *temp* files in the */usr* directory that have not been accessed for one week (**-atime +7**), enter:

find /usr -name temp -atime +7 -print

Once you locate the *temp* files, you can remove them automatically using the **-exec** option to **find**. See "Executing commands based on find output" (this page).

Executing commands based on find output

You can execute a specific shell command on the files that **find** locates using the **-exec** option. The most common use of **-exec** is to locate a group of files and then remove them.

For example, to find all the core files in the */usr* filesystem that have not been accessed in seven days and remove them, enter:

find /usr -name core -atime +7 -exec rm "{}" \;

As another example, when you retire a user, use **find** to locate all the files owned by that user, back them up, and then remove them from the system. To do this, enter:

find / -user edwarda -print | cpio -ovBc > /dev/rfd0

find / -user edwarda -exec rm "{}" \;

The first command locates all the files owned by user *edwarda* and copies the files to a floppy disk archive. The second command locates the files and then removes them. For more information on copying files to an archive, see the **cpio**(C) manual page.

To specify that **find** prompt you with the command line that **find** generates before executing the shell command on each file, use **-ok** in place of **-exec**:

find / -user edwarda -ok rm "{}" \;

In this case, **find** prompts you with:

```
<rm ... /u/edwarda/billboard >?
```

To execute the command (in this case, **rm**), enter **y**. If you enter any character other than " y ", the command is not executed.

Another common use of **find** with the **-exec** option is to locate all the files that belong to a particular group and change them. For example, if a user changes groups, you can use **find** to locate and change all their files to the new group:

find / -user edwarda -exec chgrp pubs "{}" \;

You can use **find** to change the owner of a group of files. For example, if you retire a user and you want to transfer ownership of their files to another user, use this command:

find / -user edwarda -exec chown earnestc "{}" \;

Using this construction to execute a command on a large group of files can be very slow because the **-exec** option forks a separate process for each file in the list. A more efficient method for doing this is to use **xargs**(C) in place of **-exec**. The **xargs** command forks fewer processes to execute the command on the entire group of files.

NOTE Improper use of **find** with the **xargs** command can compromise system security. For this reason, *root* should not use **find** with **xargs**; use the **-exec** option to **find** instead.

The following example illustrates how to use the **xargs** construction with **find**:

find / -user edwarda -print | xargs chown earnestc

This command accomplishes the same thing as the previous example, only much more efficiently.

NOTE If the syntax for the command that you want to execute with **xargs** deviates from the standard order (***command options arguments***), you must use **-exec**.

Checking and clearing system log files

Your SCO system maintains a number of log files that contain information about system usage. When new information is generated, the system appends it to the appropriate log file, preserving the previous contents of the file. Because some log files can rapidly become quite large, you should check these files periodically and, if necessary, clear them by deleting their contents.

You can manage these files by:

- Using the **System Logs Manager** (page 92)
- Clearing system log files from the command line (page 92)
- Clearing log files automatically (page 93)

Table 2-4 (this page), lists the log files that are most likely to need clearing or trimming. Your system might include different log files from those listed, depending on your configuration and the utilities and application software installed. Depending on your system activity, you might need to check the files more or less frequently than indicated in the table. Use **find**(C) to locate unlisted large log files (page 88).

Table 2-4 Administrative log files

Log File	Purpose	Checking frequency
/etc/wtmp	historical login record	automatic*
/usr/adm/pacct	process accounting log file	weekly
/usr/adm/messages	system messages log file	weekly
/usr/adm/sulog	**su**(C) log file	automatic*
/tcb/audittmp/*	audit system temporary files	weekly
/usr/spool/uucp/LOGFILE	records of UUCP job requests, file transfers, system status	monthly
/usr/spool/uucp/.Log/.Old/*	old UUCP log files stored by uudemon.clean	monthly
/usr/spool/lp/logs/requests	record of print requests	automatic*

* You can maintain this file (and others) automatically; see "Clearing log files automatically" (page 93).

Using the System Logs Manager

You can examine, clear, or print log files using the **System Logs Manager**, located in the *System/Logs* directory of the SCOadmin hierarchy. The **System Logs Manager** displays a list of current log files and the contents of the selected file.

To display the contents of a log file, select the file from the list.

To search for a pattern in the log file, enter the pattern in the "Search" field. Use the **next** and **prev** buttons to search for other occurrences of the pattern.

To print out a log file, select the file from the list, then select **Print** from the **File** menu.

To add a new log file to the list, select **Include Log** from the **Log** menu. You are asked to supply a location and a comment.

To omit a log file from the list, select the file from the list, then select **Exclude Log** from the **Log** menu.

To clear the contents of a log file, select the file from the list, then select **Clear Log** from the **Log** menu.

See also:

- Table 2-4, "Administrative log files" (page 91)
- "Clearing system log files from the command line" (this page)
- "Clearing log files automatically" (page 93)

Clearing system log files from the command line

To clear a log file and retain the file permissions:

1. Copy the file to a new filename. For example:

   cp /usr/adm/messages /usr/adm/messages.old

2. Clear the file with one of these commands:
   - Bourne or Korn shell:

     > /usr/adm/messages

   - C shell:

     cat /dev/null > /usr/adm/messages

To monitor new information that is currently being appended to system log files, enter:

tail -f /usr/adm/messages

The **-f** (follow) option to the **tail**(C) command prints the last 10 lines of the file, followed by any lines that are appended to the file between the time you initiated and stopped (with ⟨Del⟩) the **tail** command.

Clearing log files automatically

You can clear log files automatically using **crontab**(C) and **cron**(C). By adding a line to *root*'s *crontab* file (*/usr/spool/cron/crontabs/root*), you can make your own file maintenance scripts execute daily, weekly, or monthly.

The following is one of the default entries in *root*'s *crontab* file:

```
17 5 * * 0 /etc/cleanup > /dev/null
```

See **crontab**(C) manual page for complete information on the format of the *crontab* file.

This entry runs **/etc/cleanup** each Sunday morning at 5:17.

NOTE If multiple machines mount the same NFS filesystems, all machines will run the **find** at the same time, searching all the NFS-mounted filesystems as well as the local filesystems. Therefore, running this script without some modifications could impose heavy network load. For example, adding the **-mount** or **-local** options to the **find**(C) command will restrict the search to the root filesystem, or local filesystems, respectively.

The **/etc/cleanup** script looks like this:

```
:
#
# clean up super-user log
cp /usr/adm/sulog /usr/adm/Osulog
> /usr/adm/sulog
#
# clean up volcopy log
[ -f /etc/log/filesave.log ] && mv /etc/log/filesave.log /etc/log/Ofilesave.log
> /etc/log/filesave.log
chown root /etc/log/filesave.log
chgrp sys /etc/log/filesave.log
chmod 666 /etc/log/filesave.log
#
# clean up wtmp
> /etc/wtmp
#
# clean up miscellaneous files
find / -name core -atime +7 -exec rm -f {} \;
```

The **/etc/cleanup** script:

- copies the *sulog* file to *Osulog* and then clears *sulog*.

- moves *filesave.log* (if it exists) to *Ofilesave.log*, creates a new, empty *filesave.log*, and sets the proper permissions.
- clears the */etc/wtmp* file.
- removes all *core* files that have not been accessed in the last seven days.

You can specify different files to clear and when to clear them by modifying **/etc/cleanup** and the */usr/spool/cron/crontabs/root* file.

Adding disk space and restructuring filesystems

A chronic shortage of space usually results from having more users on the system than the current hard disk can reasonably handle, or simply having too many directories or files. In either case, creating a new filesystem on a new hard disk allows you to transfer some of the users and directories from the primary hard disk to the new location, freeing a significant amount of space on the existing filesystem and improving system operation.

If free space is chronically low on your system, you might want to expand your system storage capacity by installing additional hard disks. See Chapter 17, "Adding hard disks" in the *SCO OpenServer Handbook*. Once you install the new hard disk and create new filesystems, you can use the free space in the new filesystem for new work, or you can copy existing user or system directories to it.

If one filesystem is full and other filesystems have a significant amount of free space (or there is additional unused space on the hard disk), you can change the layout on the primary disk to take advantage of the free space. However, this procedure is not as simple as adding a second hard disk.

To change the number of filesystems on your hard disk or to reapportion the disk space among the filesystems:

1. Create a complete backup of each filesystem on your system. See "Running unscheduled filesystem backups" (page 119).

   WARNING Be certain that you have a complete, accurate, and readable backup for each filesystem before you continue or you may lose your files.

2. Reinstall your system as described in "The installation and upgrade procedure" in the *SCO OpenServer Handbook*.

   NOTE Make sure you select customization of your hard disk layout and redistributon of disk space among the filesystems.

3. Restore the data from the filesystem backups. See "Restoring files or directories from backup media" (page 132).

See also:

- Chapter 17, "Adding hard disks" in the *SCO OpenServer Handbook*
- "Partitioning a hard disk using fdisk" in the *SCO OpenServer Handbook*
- "Dividing a disk partition into divisions using divvy" in the *SCO OpenServer Handbook*
- **fdisk**(ADM) manual page
- **divvy**(ADM) manual page

Moving a subdirectory to another filesystem using symbolic links

Another way of expanding a filesystem is to use space located in another filesystem. This can be accomplished using symbolic links. For example, if you are running out of space on the root filesystem for an application data files directory, you can use space located on a secondary filesystem. As far as the application is concerned, the files will still appear to reside in the root filesystem.

The procedure described here moves a subdirectory to another filesystem and creates a symbolic link using this example:

Subdirectory name:	subdir
Original location (path) of subdir tree:	/usr
New location (path) of subdir tree:	/d1

NOTE The original filesystem must be of type EAFS, HTFS, or DTFS.

1. Suspend operations on your system and bring it down to single-user mode.
2. Make a backup of your system, and verify the backup to make certain it is valid.
3. Manually mount the secondary filesystem(s).

   mount /dev/d1 /d1

4. Get current permissions of /usr/subdir:

   ls -ld /usr/subdir

5. Create the new subdirectory in /d1:

   mkdir /d1/subdir

6. Change destination directory ***owner, group, permissions*** to match the original source:

   chown ***owner*** /d1/subdir
   chgrp ***group*** /d1/subdir
   chmod ***permissions*** /d1/subdir

7. Copy */usr/subdir* to */d1/subdir* using these commands:

   cd /usr/subdir
   find . -depth -print | cpio -pdmv /d1/subdir

   NOTE **cpio**(C) is used instead of a copy command to better retain permissions.

8. Verify that the copy is successful:

   dircmp /usr/subdir /d1/subdir | tee /tmp/dirlog

9. Create the symbolic link:

   ln -s /d1/subdir /usr/subdir

The procedure is now complete. Whenever you change directory to */usr/subdir*, you will actually be in */d1/subdir* (assuming filesystem *d1* is mounted).

NOTE Utilities that deal with the filesystem directory structure may not function as desired unless additional options are used. These utilities include (but are not limited to): cd, ls, pwd, find, cpio, and tar. For example, if enter:

ls -ld /usr/subdir

you see the symbolic link that references the */d1/subdir* directory. To see the physical directory entry itself, with its owner, group and permission information, include the **-L** option:

ls -ldL /usr/subdir

See also:

- **cpio**(C)
- **mkdir**(C)
- **ln**(C)

Maintaining filesystem efficiency

Three aspects of filesystem usage can degrade the efficiency of filesystems.

NOTE With the exception of limiting directory size, most of the information that follows does not apply to HTFS and DTFS filesystems and is marked accordingly.

Disk fragmentation	Disk fragmentation is the scattering of available disk space caused by constant use and reuse of filesystem blocks. See "Reducing disk fragmentation" (this page).
Excessively large directories	Directories that contain large numbers of files (regardless of file size) increase the system search time for files within the directory. See "Monitoring and limiting directory sizes" (page 98).
Empty directory slots	Empty directory slots, caused by large numbers of files being created and moved or removed from a directory can also cause the directory to become too large (EAFS, AFS, and S51K filesystems only). See "Removing empty directory slots" (page 99).

Reducing disk fragmentation

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, thus scattering the available disk space. Disk fragmentation increases access time and reduces filesystem efficiency.

NOTE HTFS and DTFS filesystems attempt to cluster allocations for a particular file together on the disk, so disk fragmentation is less of a concern with these filesystem types.

To reduce disk fragmentation on EAFS, AFS, and S51K filesystems:

1. Create a complete backup of all the files in the filesystem. See "Running unscheduled filesystem backups" (page 119).

   WARNING Be certain that you have a complete, accurate, and readable backup before you continue or you may lose your files.

2. Remove all the files from the hard disk.
3. Restore the files from the backup. See "Restoring files or directories from backup media" (page 132).

This procedure rewrites all the files during the restore. Each file is written in one piece on the disk, thus consolidating the available disk space and reducing disk fragmentation. You also recover a small amount of space when you run this procedure.

Run this procedure about once a year on a heavily used system (less often on a lightly used system).

Monitoring and limiting directory sizes

To improve performance, limit the number of files and the filename length in critical directories. Limit working directories, such as login directories, to 62 files (plus the required "." and ".." entries). A directory that contains fewer than 62 files fits in a single disk block and can be searched very efficiently. Data storage directories, such as spool directories, can contain up to 638 entries and still be viable.

NOTE These directory size figures apply to filenames of 14 characters or fewer. As filename lengths increase (up to a maximum of 255 characters for most filesystem types), the number of files that fit on a single disk block decreases, thus reducing the maximum number of files in a directory.

To determine if a directory is too large, enter:

```
l . | wc -l
```

This command gives you the number of files in the directory listing. If this number is greater than 62 (for a working directory) or 638 (for a spool directory), the number of files in the directory might be affecting system performance.

To enhance performance, create a hierarchy of subdirectories with 5 to 10 directories at each level, then store the files in these subdirectories. Educate users to keep their login directories small by setting up a simple, clear hierarchy of subdirectories.

Even if you delete files so that you have fewer than 62 (or 638), the directory might still be oversized and inefficient. The directory remains the same size because empty directory slots are left when you move or remove files. Thus, if large numbers of files are moved in and out of a directory (as in the case of a spool directory), the directory can remain very large even if there are very few files in the directory.

You can reduce the size of the directory by reducing the number of empty directory slots (this page).

Removing empty directory slots

When files are deleted or moved to subdirectories in EAFS, AFS and S51K filesystems, "empty directory slots" (or "shadow files") are left. These empty slots can affect system performance by slowing down directory searches.

NOTE In HTFS and DTFS filesystems, empty directory slots are merged automatically, so it is not necessary to remove them. Attempts to use the **hd**(C) command as described in this section will fail on HTFS and DTFS filesystems.

To determine if a directory contains empty slots, enter:

```
hd . | wc -l
l | wc -l
```

These commands display the number of inodes and number of files in the directory listing. If the number of inodes is significantly more than the number of files in the directory, your system performance might be affected.

To view empty directory slots, enter:

```
hd .
```

You see a listing like this:

```
0000    8c 17 2e 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
0010    4e 1d 2e 2e 00 00 00 00  00 00 00 00 00 00 00 00   N...............
0020    00 00 54 4d 2e 32 33 35  37 35 2e 30 30 31 00 00   ..TM.23575.001..
0030    00 00 44 2e 63 68 6f 72  2d 61 63 35 38 39 66 31   ..D.chor-ac589f1
0040    00 00 58 2e 63 68 6f 72  2d 75 73 41 61 63 35 38   ..X.chor-usAac58
0050    00 00 44 2e 63 68 6f 72  2d 61 63 35 36 34 39 35   ..D.chor-ac56495
0060    00 00 58 2e 63 68 6f 72  2d 75 73 41 61 63 35 36   ..X.chor-usAac56
0070    00 00 43 2e 63 68 6f 72  2d 75 73 4e 33 32 32 61   ..C.chor-usN322a
0080    00 00 41 2e 63 68 6f 72  2d 75 73 4e 33 32 32 39   ..A.chor-usN3229
0090
```

The shadow files appear at the end of the list, and have null inode numbers; 00 and 00 appear in the first two columns of hex digits. In the example, all but the first two files are shadow files.

To remove these empty file slots from the directory:

1. Change to the directory above the oversized one and create a backup directory. For example, to remove the empty slots from a spool directory (in this case, */usr/spool/uucp/pdxbox*), enter these commands:

   ```
   cd /usr/spool/uucp
   mkdir pdxbox.old
   ```

2. Move to the oversized directory and back it up with **cpio**(C):

   cd pdxbox
   find . -print | cpio -pdlm ../pdxbox.old

   This command copies the directory and all subdirectories to the backup directory, linking files instead of copying them where possible.

3. Confirm that the files were copied and that the ownership, group, and permissions of the new directory and files match the original:

   cd ..
   l pdxbox pdxbox.old

4. Use **hd** to check the backup directory to verify that all the empty slots are gone:

   hd pdxbox.old

   The output of this command should not list any files with null inode numbers (00 in the first two columns).

5. Delete the original, oversized directory:

   rm -rf pdxbox

6. Rename the backup directory to replace the oversized one:

   mv pdxbox.old pdxbox

The directory appears the same, but you can search it more quickly.

You can improve system performance by using this procedure on other working and spool directories.

See also:

- "Displaying filesystem and directory usage statistics" (page 86)
- "Monitoring and limiting directory sizes" (page 98)
- "Out of inodes on filesystem" (this page)

Out of inodes on filesystem

An inode is an internal structure that the operating system uses to track and control information about a file, such as size and last access date. Each file uses one inode. When a filesystem runs out of inodes, the system displays the following error message:

`NOTICE: `***type***`: Out of inodes on `***type***` dev hd (`***major/minor***`)`

where ***type*** is the filesystem type and ***major/minor*** is the major and minor device numbers (for example `hd (1/42)` for the root filesystem).

To fix this problem:

1. Remove unnecessary (old, temporary, *core*, or log) files from the filesystem. See "Maintaining free space in filesystems" (page 85).
2. Use **find**(C) to determine whether the filesystem contains a large number of small files (page 88).

   The initial allocation of inodes assumes a ratio of about four data blocks per inode. If the filesystem contains mostly files that are smaller than four blocks, it runs out of inodes.

The number of inodes available on a filesystem is determined when you create the filesystem using **mkfs**(ADM). If the filesystem consistently runs out of free inodes, you can reconfigure the filesystem and increase the number of inodes.

WARNING This procedure destroys the information on your hard disk. Do not use **mkfs** without first creating a complete and verified backup.

To increase the number of available inodes:

1. Back up the filesystem and verify the integrity of the backup using the **Backup Manager**. See "Running unscheduled filesystem backups" (page 119).
2. Unmount the filesystem. See "Mounting and unmounting filesystems" (page 74).
3. From the command line, run **mkfs**(ADM) and specify more inodes for the filesystem. For example, to reconfigure the number of inodes on the */dev/u* filesystem to 6400, enter:

   mkfs /dev/u ***fssize*:6400**

   Replace ***fssize*** with the size of the filesystem (in 512-byte blocks). See the **mkfs**(ADM) manual page for more information.
4. Mount the filesystem. See "Mounting and unmounting filesystems" (page 74).
5. Restore the filesystem from the backup using the **Backup Manager**. See "Restoring files or directories from backup media" (page 132).

Troubleshooting the Filesystem Manager

Most error messages displayed by the **Filesystem Manager** are self-explanatory. In other cases, the error message will be generic, for example:

```
Could not get authorization data for Filesystem Manager

Error while mounting /filesystem
```

Error boxes of this type include a **Details** button to provide you with additional information. The problems reported fall into these categories:

- Remote administration problem (this page)
- Missing or corrupted database files (page 103)

Remote administration problem

If you are performing remote administration, the remote system may be unreachable or there may be a configuration problem (including the lack of user equivalence):

localhost `failed to connect to remote host... Error with server process: Permission denied.`
These messages (and suggested solutions) are displayed when an **Open Host** operation fails. Check to make certain you have user equivalency on the remote host. See "Adding user equivalence" in the *Networking Guide* for more information.

`mount:` ***host:/filesystem*** `server not responding: RPC: Program not registered`
The filesystem could not be mounted. Check to see if NFS is running on the remote host.

`mount: access denied for` ***host:/filesystem***
The host is not available or the filesystem has not been exported.

`mount:` ***host*** `not in hosts database`
The remote host is not listed in */etc/hosts* or is no longer recognized by the nameserver. Check that the nameserver is running.

`umount:` ***/filesystem*** `not mounted: Invalid argument`
The filesystem is not currently mounted and the manager is attempting to mount it. Cancel the action and try again.

`Error while [un]mounting` ***filesystem***
The remote filesystem is no longer available. Check the remote host and see if the network is up and running and that the filesystem is exported.

For more information on network problems, see "Troubleshooting TCP/IP" in the *Networking Guide.*

Missing or corrupted database files

The **Details** may indicate file data (or the file itself) is missing, as in these examples:

`mnt: cannot open /etc/default/filesys`
The file */etc/default/filesys* is missing. Restore it from backups.

`Failed to remove export configuration for` ***filesystem*** `- the object instance` ***filesystem*** `does not exist`
The file */etc/exports* is missing. Restore it from backups.

`Failed to remove mount configuration for` ***filesystem***
The file */etc/default/filesys* is missing. Restore it from backups.

Chapter 3

Backing up filesystems

The main task of a system administrator is ensuring the continued integrity of information stored on the system. One way the system administrator maintains integrity is to back up the data on the system periodically so that the data can be restored if it is lost. The **Backup Manager** (page 106) enables the system administrator (*root* or a user with **backup** and **sysadmin** authorization) to save a copy of a filesystem.

NOTE All local filesystems in the */etc/default/filesys* file are listed in the **Backup Manager**, whether the filesystem is scheduled to be backed up or not. No mounted NFS filesystems are listed. To back up an NFS filesystem, open the host where the filesystem resides as described in "Selecting another host to manage" in the *Networking Guide* and back it up there, or add the remote filesystem (page 126) to be backed up from the local host.

Before you use the **Backup Manager** to back up your system:

1. Set the default backup device (page 135).
2. Set the block and volume size for the backup device (page 136).
3. Add the backup schedule for each local filesystem (page 121).
4. Add the backup schedule for any remote filesystems (page 126).

Once you set it up, use the **Backup Manager** to:

- run a scheduled backup (page 112).
- run an unscheduled backup (page 119).
- restore a filesystem from scheduled backups (page 129).
- restore files or directories from backup media (page 132).

- examine the contents of a backup (page 129).
- examine the backup history for a filesystem (page 127).

See also:

- "About backups" (page 107)
- "Performing unattended backups" (page 117)
- "Specifying the Backup Manager default values" (page 135)
- "Assigning subsystem authorizations" (page 32)
- "Using the command line to create and restore backups" (page 136)

The Backup Manager interface

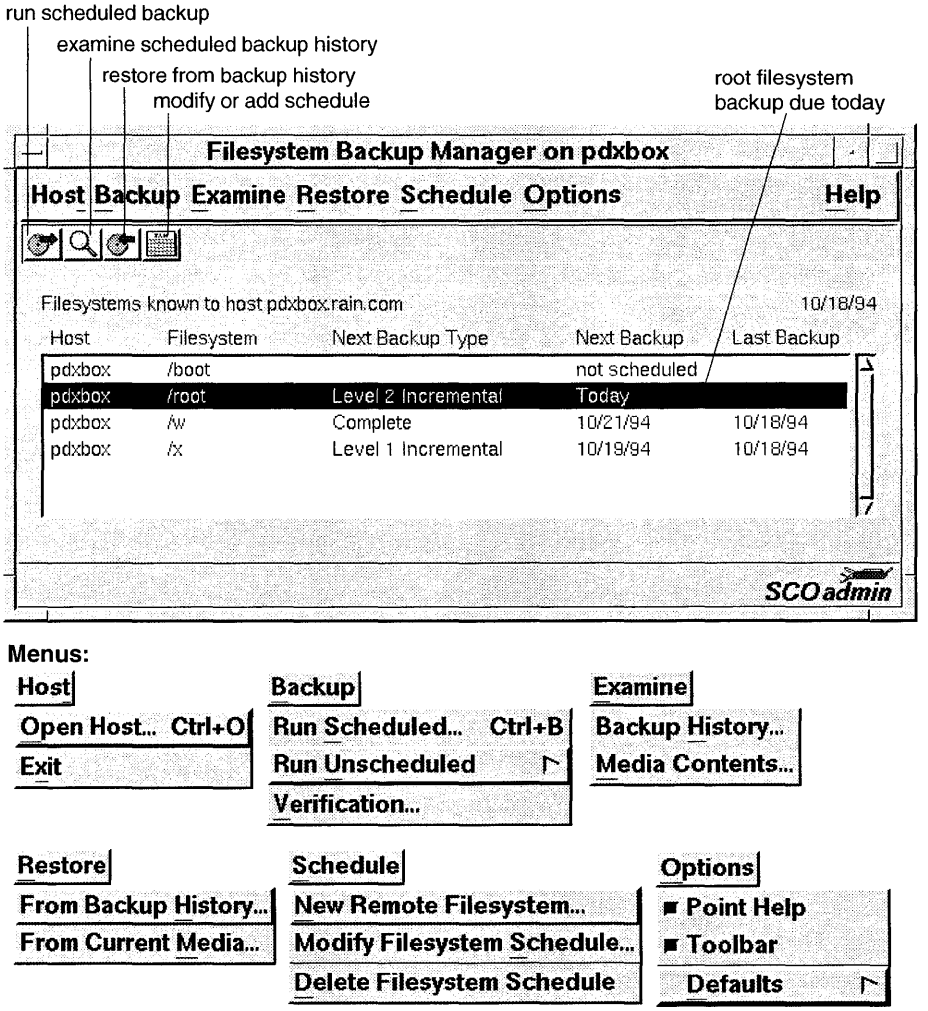
Use the **Backup Manager** to back up and restore filesystems. You can start the **Backup Manager** in any of these ways:

- Double-click on the **Backup Manager** icon in the *System Administration* window on the Desktop.
- Start the SCOadmin launcher by entering **scoadmin** on the command line, then selecting **Backup Manager**.
- Enter **scoadmin backup manager** on the command line (or abbreviate to **scoadmin b**).

For more information on using SCOadmin managers, see "Administering your system with SCOadmin" in the *SCO OpenServer Handbook*.

Authorization

When run from an account other than *root*, use of the **Backup Manager** requires the **backup** authorization. In addition, backups may involve mounting filesystems, which requires the **sysadmin** authorization. See "Assigning subsystem authorizations" (page 32) for more information.



About backups

A "backup" is a copy on external storage media (floppy disks or tape), of files on your hard disk. A "filesystem backup" is a copy of the files in the *root* filesystem or another regularly mounted filesystem (such as */u*). See Chapter 2, "Administering filesystems" (page 53) for more information about filesystems.

Files and filesystems can be damaged and data can be lost through:

- power surges and interruptions (make certain you have a surge protector).
- hardware failures (particularly the hard disk).
- user errors (accidental removal of important files).

You should back up your filesystems regularly so you have up-to-date copies from which to restore lost data. Use the **Backup Manager** to back up and restore filesystems.

NOTE You must be logged in as *root* or have **backup** authorizations to create backups with the **Backup Manager**. If the filesystem is unmounted prior to running a backup, you must be logged in as *root* or also have **sysadmin** authorizations to mount the filesystem. See "Assigning subsystem authorizations" (page 32).

A "scheduled backup" is a regular backup performed according to a predefined schedule. The backup schedule determines the days of the week to perform the backup, which backup level to perform on which days, and so forth. When you use a backup schedule, you do not need to save the entire filesystem each time you perform a backup. Instead, you can perform an "incremental backup", which saves only those files that have changed since a specified time. See "Understanding incremental backups" (page 124).

If your system has many users and a large number of files that are modified daily, you should create a backup schedule and perform regularly scheduled backups. To schedule a local filesystem to be backed up regularly, see "Adding, modifying, and removing filesystem backup schedules" (page 121). To schedule backups for a remote filesystem, see "Adding remote filesystems to the backup schedule" (page 126).

NOTE To back up remote filesystems with the **Backup Manager**, you must first establish *backup* user equivalence (page 126).

An "unscheduled backup" is a complete, informal backup of a filesystem or directory. Unscheduled backups copy the entire filesystem or directory, not just the modified files, and thus might require many storage media volumes. If your backup requirements are simple, you might be able to perform unscheduled backups on a regular basis. (we recommend at least once a week).

See also:

- "Running unscheduled filesystem backups" (page 119)
- "Running scheduled backups" (page 112)

About media devices

The **Backup Manager** uses the Device Query Interface (DQI) to determine what media devices are available on your system and display their names wherever you are prompted to select a media device. You can also use a device that is not supported in the DQI by deleting the current entry in the "Media Device" field and entering the desired device file (for example, */dev/rct0*).

NOTE To permanently add a non-DQI device to the list of default devices, see "Setting the default media values" (page 136).

Depending on the hardware you have installed, the **Backup Manager** supports these media devices by default:

Table 3-1 Media devices

Device	Description	Device File
1/4-inch Cartridge Tape Drive 0	Primary cartridge tape	/dev/rct0
1.44MB 3.5in Floppy Drive 0	Primary 1.44MB floppy	/dev/rfd0135ds18
1.44MB 3.5in Floppy Drive 1	Secondary 1.44MB floppy	/dev/rfd1135ds18
720KB 3.5in Floppy Drive 0	Primary 720KB floppy	/dev/rfd0135ds9
720KB 3.5in Floppy Drive 1	Secondary 720KB floppy	/dev/rfd1135ds9
360KB 5.25in Floppy Drive 0	Primary 360KB floppy	/dev/rfd048
360KB 5.25in Floppy Drive 1	Secondary 360KB floppy	/dev/rfd148
1.2MB 5.25in Floppy Drive 0	Primary 1.2MB floppy	/dev/rfd096
1.2MB 5.25in Floppy Drive 1	Secondary 1.2MB floppy	/dev/rfd196

To set a specific media device to be the default, see "Setting the default backup device" (page 135).

For each media device on your system, the **Backup Manager** uses default values for the block and volume sizes (this page). To change these default values, see "Setting the default media values" (page 136).

NOTE Although the DQI automatically senses the type of media on your system, in some cases the DQI cannot automatically sense the capacity (volume size) of the drive, and so sets the volume size to 0. To use the **Backup Manager** to back up such a filesystem, you must first set the volume size for the devices on your system. See "Setting the default media values" (page 136) for more information.

See also:

- "Preparing media for backups" (page 111)
- **dat**(HW) manual page
- **tape**(HW) manual page

About block and volume sizes

Block size the size of each block (in bytes) written on the media. If the files to back up are small, set the block size to a smaller value. If you set the block size to a larger value, the access time is less efficient.

Volume size the capacity of the media (in KB). For example, a cartridge tape might be capable of storing 60MB (60000KB), 150MB (150000KB), or 520MB (520000 KB).

WARNING Do not change the default volume sizes for floppy drives.

Specify block and volume size in bytes.

See also:

- Note about DQI and volume size (this page)
- "About media devices" (page 109)
- "Setting the default media values" (page 136)

Preparing media for backups

Floppy disks and some types of cartridge tapes require preparation before you can create backups on them.

Before using floppy disks with the **Backup Manager**, you might need to use the **format**(C) command to format them.

You should format all new floppy disks and any disks formatted under another operating system before using them on your SCO system. Once you format disks, you can use them over again without reformatting. However, disks formatted under some operating systems cannot be used under other operating systems, even with reformatting.

WARNING Any data on the disk is lost when you format it.

Before using a cartridge tape with the **Backup Manager**, you might need to prepare it. Use the **tape**(C) command to:

erase Erase and retension the tape cartridge.

format Format the tape. You must format floppy-controller-based tapes before using them. Before reformatting a used tape, you must erase it with a bulk tape eraser.

NOTE Do not format Irwin tape cartridges. Use preformatted cartridges for best results.

reten Retension the tape periodically to fix slack tape problems. Tape slack can cause unusually large numbers of tape errors.

rewind Rewind to the beginning of the tape.

If you use tape cartridges that require formatting, you should format the volumes before beginning the backup. 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 Chapter 7, "Working with disks, tapes, and CD-ROMs" in the *Operating System User's Guide*.

See also:

- "About media devices" (page 109)
- "Setting the default media values" (page 136)
- **tape**(C) manual page
- **format**(C) manual page (for command-line interface)

Running scheduled backups

NOTE Before performing a scheduled backup, you might need to format the media (page 111).

In the **Backup Manager** (page 106):

1. Select the filesystem to back up from the list.

   NOTE If you select a remote host to back up, verify that the system time on the local and remote hosts do not differ by more than one day. If the system times are off by more than one day, the backup may fail.

2. Select **Run Scheduled** from the **Backup** menu.

   NOTE Unless a filesystem is scheduled to be backed up today or overdue ("Today" or "Overdue" appears in the "Next Backup" column), the **Run Scheduled** item is inactive (dimmed).

3. If necessary, select or enter the correct "Media Device".
4. If necessary, click on the **Media Options** button to change:
   - "Block Size" (page 110)
   - "Volume Size" (page 110)
5. Click on **Start**.
6. When prompted, insert each media volume.
7. Verify the backup by clicking on **Yes** at the prompt, inserting the requested backup volume, and clicking on **OK**.

   To skip the verification process, click on **No**.
8. When the backup is complete, click on **OK**.
9. Label your backup and store it in a safe place.

See also:

- Note about DQI and volume size (page 110)
- "About the backup schedule" (page 123)
- "Performing unattended backups" (page 117)
- "Maintaining backup archives and records" (page 113)

Maintaining backup archives and records

After creating your scheduled backups, store them in a secure, fireproof place so that you can use them to restore your data in the event of a hardware failure or other system problem that causes data loss. Maintaining backup archives involves:

- "Labeling backups" (this page)
- "Keeping a backup log book" (page 114)
- "Rotating and archiving backup media" (page 114)
- "Removing file lists from the backup history" (page 115)

Labeling backups

Label your backup media with meaningful and accurate information so you will be able to locate your data easily at a later date.

Example 3-1 Sample media label

Computer: pdxbox	Backup Level: 0 (Complete) Filesystem: /usr Save Until: 30 Dec 1995	Date: 30 Dec 1994
Performed by: elvis		Volume 1 of 3

The date on the label, and the date from which you calculate the "Save Until" date, should be the date of the business day last covered by the backup. This is to avoid confusion if it becomes necessary to restore information from this tape.

Color-coding labels can help you easily locate a backup by backup level.

Table 3-2 Sample backup label color scheme

Backup level	Volume size	Save for how long	Vitality (importance)	Label marker
0 (Complete)	-	1 year	critical	red sticker
1	-	4 months	necessary	yellow sticker
2	-	3 weeks	useful	blue sticker

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.

See also:

- "Understanding incremental backups" (page 124)
- "Keeping a backup log book" (this page)
- "Rotating and archiving backup media" (this page)

Keeping a backup log book

We recommend that you maintain a written log book for each computer at your site. Record all the information about the hardware and software configuration for each computer, as well as all maintenance information (such as when a breakdown occurs and what was done to correct the problem).

In addition, use this log book to keep track of the backup history for each computer. If your system is damaged such that the online backup history information is unavailable, you can use this information to construct a backup set from which to restore your system.

For each backup of the system:

Date Last day included in the backup.

Filesystem Name of the filesystem backed up (such as "*/usr*").

Backup level Level of the current backup (such as "Level 2").

 # Volumes Number of tape volumes in the backup.

Start/Finish Time (optional)
Time from the start of a backup of a filesystem until the last error check is completed.

If there are problems with the backup, record these in the log book as well, including any error messages that occur.

See also:

- "Understanding incremental backups" (page 124)

Rotating and archiving backup media

You should retain backups so that you have between 6 and 12 months of media on file. The typical media rotation times are:

Backup level	Save for
2	3 weeks
1	4 months
0 (Complete)	1 year

This means that if you follow this schedule, you can safely reuse your Level 2 backup media after 3 weeks.

You should periodically archive your filesystem backups offsite. In the event of a fire or other catastrophe, you can use these offsite backups to restore your data.

See also:

- "Removing file lists from the backup history" (this page)
- "Labeling backups" (page 113)
- "Understanding incremental backups" (page 124)

Removing file lists from the backup history

If you specified that the backup contents lists (page 128) be saved, the **Backup Manager** automatically saves the names of the files and directories that were backed up each time you perform a scheduled backup.

You should retain these file lists online for the backup media that you archive. When you rotate your media, you should also remove the file lists from your hard disk.

To remove these file lists automatically, add an entry in the *root crontab* file to specify that **cron**(C) run the **bshrink**(ADM) utility at the desired times. The **bshrink** utility removes the stored file lists from the backup history after they have aged a specified number of days; **bshrink** does not remove any other backup history records.

The syntax for **bshrink** is:

bshrink [**-h** *hostname*] "*filesystem backup_level days*" [...]

hostname	the host to manage (optional). Unless specified, **bshrink** removes files from the local host.
filesystem	the filesystem device name (for example, */dev/u*).
backup_level	the backup level:
	COMPLETE Level 0 backup
	INCR1 Incremental Level 1 backup
	INCR2 Incremental Level 2 backup
days	an integer defining the number of days to save the backup file lists for the given filesystem and level.

To exclude a filesystem name or backup level, precede it with the keyword NOT. To include all filesystems or backup levels, replace the option with the keyword ANY.

If you use the media rotation schedule in "Rotating and archiving backup media" (page 114), use this schedule to remove the backup lists from your hard disk:

Backup level	Remove after	bshrink options
2	3 weeks	"ANY INCR2 21"
1	4 months	"ANY INCR1 120"
0 (Complete)	1 year	"ANY COMPLETE 365"

Once you decide on a backup list removal schedule, add a line to the */usr/spool/cron/crontabs/root* file to run **bshrink** according to your schedule.

For example, to run **bshrink** at 2:00am every morning on the local host, add this **bshrink** line:

```
0 2 * * * /usr/bin/bshrink "ANY COMPLETE 365" "ANY INCR1 120" "ANY INCR2 21"
```

In this case, **bshrink** removes file lists from the local host for:

- Level 0 (Complete) backups older than 365 days (1 year) of any filesystem.
- Level 1 backups older than 120 days (4 months) of any filesystem.
- Level 2 backups older than 21 days (3 weeks) of any filesystem.

To run **bshrink** at 5:00 pm every Sunday morning on the host *pdxbox*, add this **bshrink** line:

```
0 17 * * 7 /usr/bin/bshrink -h pdxbox "ANY COMPLETE 365" "ANY INCR1 120" "ANY INCR2 2
```

NOTE For each host from which you want to remove file lists, you must specify a separate **cron** entry.

See also:

- **bshrink**(ADM) manual page
- **crontab**(C) manual page
- **cron**(C) manual page
- "Understanding incremental backups" (page 124)
- "About the backup history" (page 127)

Verifying backups

To ensure that your backup volumes are accurate and error-free, use the **Backup Manager** to verify the backup.

NOTE The **Backup Manager** always prompts you to verify the backup immediately after performing a backup.

In the **Backup Manager** (page 106):

1. Select **Verification** from the **Backup** menu.
2. If necessary, select or enter the correct "Media Device" (page 109).
3. When prompted, insert the media in the drive for each volume,

   The **Backup Manager** checks the volume to verify that it is readable.
4. Once the volume is verified, click on **OK** to continue.

Performing unattended backups

Because running backups can slow down the system, you might find it more convenient to perform backups when the system load is low, for example, during early mornings. You can set up your system to run backups when the system is unattended.

NOTE A single day's backups must fit on one volume for unattended backups to succeed.

To set up your system for incremental unattended backups, add an entry in the *root crontab* file to specify that **cron**(C) run the **cbackup**(ADM) shell script at the desired times. Only *root* can run **cbackup** (directly or from **cron**); users with **backup** subsystem authorizations cannot run **cbackup**.

Using **cbackup** bypasses the backup schedule for a particular filesystem, and calls the **cpio**(C) utility directly to perform the backup. However, **cbackup** still allows you to make incremental backups. **cbackup** takes these arguments and passes them to **cpio**:

Argument	Description
level	backup level to perform (0, 1, or 2)
len	capacity of the backup media volume (in KB)
device	device name on which to record the backup, for example, */dev/rct0* for a cartridge tape
filesystem	device name of the filesystem to back up, for example, */dev/root*

NOTE **cbackup** looks up the mount point of the filesystem in */etc/default/filesys*; because **cbackup** uses **cpio**, the filesystem must be mounted when it is backed up.

For example, to run a complete (Level 0) backup of the *root* filesystem on a 150MB cartridge tape at 2:00am every morning, edit the */usr/spool/cron/crontabs/root* file and add the following **cbackup** line:

```
0 2 * * * /usr/lib/sysadmin/cbackup 0 150000 /dev/rct0 /dev/root
```

Even though **cbackup** bypasses the backup schedule, you can construct a hierarchy of incremental backups by stating in the *crontab* file when to perform a particular level.

For example, to perform a Level 1 backup of the *root* filesystem at 2:00am every weekday morning and a complete (Level 0) backup on a 150MB cartridge tape on Saturday morning, include the following **cbackup** lines in the *crontab* file:

```
0 2 * * 1-5 /usr/lib/sysadmin/cbackup 1 150000 /dev/rct0 /dev/root
0 2 * * 6   /usr/lib/sysadmin/cbackup 0 150000 /dev/rct0 /dev/root
```

When you run unattended backups, remember to change the backup media in the drive between backups and label and store the incremental backups. See "Maintaining backup archives and records" (page 113).

To back up more than one filesystem on the same unattended media device (without inserting new media), run the backups sequentially. The best way to do this is to create a shell script in the */usr/lib/sysadmin* directory (for example, **bscript**) to make the calls to **cbackup**. Then, add an entry to *root*'s **crontab** file to run **bscript**. Remember, the combined total size of the filesystems you want to back up must fit on one volume for these unattended backups to succeed.

In this example, **/usr/lib/sysadmin/bscript** performs a Level 1 backup of the */u* and */w* filesystems to a 150MB cartridge tape (the size of */w* is 100MB; */u* is 50MB):

```
/usr/lib/sysadmin/cbackup 1 150000 /dev/nrct0 /dev/w
/usr/lib/sysadmin/cbackup 1 50000 /dev/nrct0 /dev/u
tape rewind /dev/xct0
```

The ***len*** argument to the first **cbackup** command is **150000** because the total capacity of the media at that time is 150MB; ***len*** is **50000** for the second call to **cbackup** because the */w* filesystem has already been backed up, reducing the total capacity of the device to 50MB.

NOTE In this example, the device is */dev/nrct0*, the no-rewind tape device. Use this device instead of */dev/rct0* so that the tape does not automatically rewind to the beginning after backing up */dev/w*. The **tape rewind** command rewinds the tape to the beginning after the backup of */u* is complete.

Now, add an entry to */usr/spool/cron/crontabs/root* to run **bscript** every weekday morning at 2:00am:

```
0 2 * * 1-5 /usr/lib/sysadmin/bscript
```

See also:

- **cbackup**(ADM) manual page
- **crontab**(C) manual page
- **cron**(C) manual page
- **tape**(C) manual page

Running unscheduled filesystem backups

NOTE Before performing an unscheduled backup, you might need to format the media (page 111).

In the **Backup Manager** (page 106), to run an unscheduled backup of a listed filesystem (filesystems known to the local host):

1. Select **Run Unscheduled** from the **Backup** menu, then select **Selected Filesystem**.
2. If necessary, select or enter the correct "Media Device" (page 109).
3. If necessary, click on the **Media Options** button to change:
   - "Block Size" (page 110)
   - "Volume Size" (page 110)
4. Verify that the name of the directory from which to start the backup is correct.
5. Click on **Start**.
6. When prompted, insert each media volume.
7. Verify the backup by clicking on **Yes** at the prompt, inserting the requested backup volume, and clicking on **OK**.

   To skip the verification process, click on **No**.
8. When the backup is complete, click on **OK**.

See also:

- Note about DQI and volume size (page 110)
- "About backups" (page 107)

Running unscheduled backups of other remote filesystems

NOTE Before performing an unscheduled backup, you might need to format the media (page 111) and establish *backup* user equivalence (page 126).

In the **Backup Manager** (page 106), to run an unscheduled backup of a remote filesystem that is not listed in the **Backup Manager** main window (a filesystem unknown to the local host):

1. Select **Run Unscheduled** from the **Backup** menu, then select **Other Remote Filesystem**.
2. Select the host on which the remote filesystem resides. See "Selecting another host to manage" in the *Networking Guide*.
3. Select the filesystem to back up.
4. If necessary, select or enter the correct "Media Device" (page 109).
5. If necessary, click on the **Media Options** button to change:
   - "Block Size" (page 110)
   - "Volume Size" (page 110)
6. Verify that the name of the directory from which to start the backup is correct.
7. Click on **Start**.
8. When prompted, insert each media volume.
9. Verify the backup by clicking on **Yes** at the prompt, inserting the requested backup volume, and clicking on **OK**.

   To skip the verification process, click on **No**.
10. When the backup is complete, click on **OK**.

See also:

- Note about DQI and volume size (page 110)
- "About backups" (page 107)

Adding, modifying, and removing filesystem backup schedules

To add or modify a filesystem backup schedule, in the **Backup Manager** (page 106):

1. Select the filesystem from the list.

   (Unscheduled filesystems show `not scheduled` in the "Next Backup" column; scheduled filesystems show a date in this column.)

2. Select **Add Filesystem Schedule** or **Modify Filesystem Schedule** from the **Schedule** menu.

3. Modify the filesystem backup schedule:

   - Change the **schedule length** by clicking on the **Up** arrow or **Down** arrow button or press ⟨Del⟩ and enter a new number. See "About the backup schedule" (page 123).
   - Change the **backup level** for each day in the schedule cycle by clicking on the "Day". For example, to change the backup level on "Day 6" from `No backup` to `Level 2 Incremental Backup`, click on "Day 6" two times. See "Understanding incremental backups" (page 124).

     When you set the backup level to something other than `No backup`, a check mark appears in the left column to indicate that a backup will be performed on that day in the schedule.
   - Specify where **Today** (the current date) falls in the backup schedule by clicking on the **Up** or **Down** arrow button or by pressing ⟨Del⟩ and entering a new number.

     For example, if you want to start by performing a complete backup of the filesystem today, specify that "Today" is Day 1. If you are already performing regularly scheduled backups and you want to perform a complete backup tomorrow, set "Today" to be the last day in the backup schedule. In other words, if you perform a complete backup every seven days (your backup schedule is seven days long), set "Today" to Day 7.
   - Specify how the filesystem is mounted and whether to save backup file lists for the filesystem by clicking on the **Backup Options** button. See "Modifying scheduled filesystem backup options" (page 122).

   When you finish setting up the backup schedule, click on **OK**.

To remove a filesystem backup schedule, select the filesystem from the main list, then select **Delete Filesystem Schedule** from the **Schedule** menu.

To modify the default backup schedule (the schedule that is used whenever you add a new filesystem to the backup schedule), select **Defaults** from the **Options** menu, then select **Schedule**. Proceed with step 3 (page 121) of the above procedure.

See also:

- "Running scheduled backups" (page 112)
- "About backups" (page 107)

Modifying scheduled filesystem backup options

In the **Backup Manager** (page 106):

1. Select the filesystem to modify.
2. Select **Modify Filesystem Schedule** from the **Schedule** menu.
3. Click on the **Backup Options** button and modify the scheduled backup options:
   - Select filesystem mount options:

Always Mount Filesystem	Select this option to mount the filesystem automatically if the filesystem is not mounted when a backup administrator runs a scheduled backup.
Never Mount Filesystem	Select this option if you do not want the **Backup Manager** to mount the filesystem. For example, select this option if the filesystem contains sensitive information and you do not want a backup administrator without *root* permissions or **sysadmin** authorization to mount the filesystem.
Prompt Operator	Select this option to prompt the backup administrator to mount the filesystem if it is not mounted.

The **Backup Manager** will not back up the filesystem unless it is mounted.

NOTE If the filesystem is unmounted prior to running a backup, you must either be logged in as *root* or have **sysadmin** authorizations to mount the filesystem.

See "Mounting and unmounting filesystems" (page 74).

- Select or deselect **Save Backup Contents Lists** (page 127).

  NOTE If you do not save the backup contents lists, the **Backup Manager** will not be able to display the online file lists when you browse the backup history (page 127) or select files to restore (page 133). The **Backup Manager** will have to read the media before displaying the file lists.

  If you select **Save Backup Contents Lists**, you should also use **bshrink**(ADM) to remove these file lists periodically. See "Removing file lists from the backup history" (page 115).

To set the backup options for the default backup schedule (the schedule that is used whenever you add a new filesystem to the backup schedule), select **Defaults** from the **Options** menu, then select **Schedule**. Proceed with step 3 (page 122) in the previous procedure.

About the backup schedule

Before performing a scheduled backup of a filesystem, you must first set up a backup schedule for that filesystem. Once you create a backup schedule, you only have to insert a media volume and respond to a series of prompts to perform your daily backups; the backup system automatically locates modified files and copies them to the backup media.

The backup schedule defines these characteristics of the scheduled backup:

- The length of the schedule cycle (how many days between performing complete backups). The default is 7 days, maximum length is 35 days.
- The backup level to use for each backup. See "Understanding incremental backups" (page 124).
- Where today (the current date) falls in the backup schedule.
- Whether to mount the filesystem if it is not mounted when the scheduled backup is run. See "Mounting and unmounting filesystems" (page 74).
- Whether to save the backup history file lists. See Saving backup contents lists (page 128).

NOTE If you are scheduling more than one large filesystem, do not schedule two complete (Level 0) backups of those filesystems on the same day. Complete backups take a long time and can slow down your machine significantly.

See also:

- "Modifying scheduled filesystem backup options" (page 122)

Understanding incremental backups

The most straightforward and dependable way to ensure the safety of the data on your system is to back up each entire filesystem every day. However, because filesystems can be large (200MB or more) this method can take hours. To make backing up files more efficient, saving both time and media, you can perform incremental (scheduled) backups instead. An incremental backup saves only those files that have changed since a previous backup, thus significantly reducing the duration and size of the backup.

To create incremental backups, the backup facility uses a progressive series of levels, each based on the last occurrence of a lower-level backup.

Table 3-3 Backup levels

Level	Backs up
Level 0 (Complete)	all files on the filesystem
Level 1	files changed since last Level 0 backup
Level 2	files changed since last Level 1 or Level 0 (Complete) backup

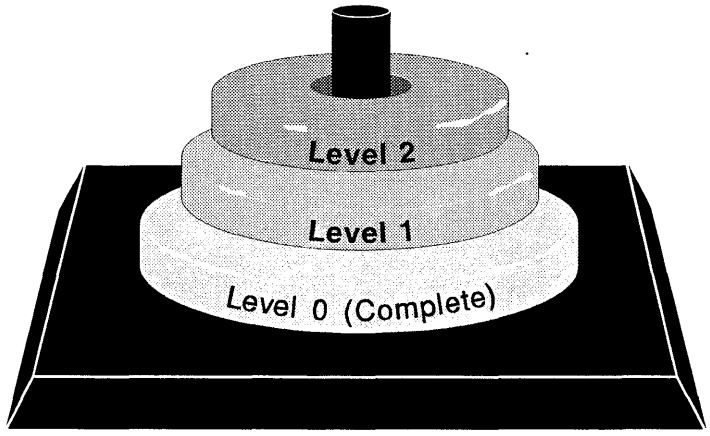


Figure 3-1 Backup levels

For example, the following backups were done for a week:

Day	Level	Files backed up
Mon	0	all files on filesystem
Tue	2	all files changed since Monday
Wed	1	all files changed since Monday
Thu	2	all files changed since Wednesday
Fri	2	all files changed since Wednesday

This example serves to demonstrate how the levels work. Each backup saves the files that have changed since the next lower-level backup. Level 0 is the lowest. Therefore, the Level 2 backup on Friday saves all files changed since the next lowest number (the Level 1 backup performed on Wednesday). The Level 2 backup on Tuesday saves only those files that have changed since the day before, because the only previous lower-level backup is a 0. If all the backup levels except Monday were Level 2, each would still back up all files that changed since the Level 0 on Monday.

Backup levels might seem needlessly complex, but consider a system with a number of large filesystems. If you performed a full backup of each filesystem each night, the process would take hours to perform, bogging the system down in the process. If the backup saves only the files that changed since the last backup, the process is less time-consuming and, depending on the size of your media, consumes fewer volumes.

Consider the following incremental backup scheme:

Monthly Perform a complete backup.

Weekly Back up everything newer than last week.

Daily Back up everything newer than yesterday.

In this scheme, 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. Each day, any files that have changed since yesterday are backed up.

To reconstruct a damaged filesystem, restore the last full (monthly) backup, the last weekly backup, and any daily backups that happened just prior to the loss of data.

See also:

- "How backups restore complete filesystems" (page 130)
- "About the backup schedule" (page 123)

Adding remote filesystems to the backup schedule

In the **Backup Manager** (page 106):

1. Select **New Remote Filesystem** from the **Schedule** menu.
2. Select the host where the filesystem is located and click on **OK**.
3. Select the filesystem (or filesystems) on the remote host to add to the schedule and click on **OK**.

   This adds the remote filesystem to the list and sets it up to be backed up on the local system using the default backup schedule.

   NOTE When you run scheduled backups of remote hosts, verify that the system time on the local and remote hosts do not differ by more than one day. If the system times are off by more than one day, the backup may fail.

4. Verify that the backup schedule is correct. See "Adding, modifying, and removing filesystem backup schedules" (page 121).
5. Establish *backup* user equivalence (this page).

See also:

- "Selecting another host to manage" in the *Networking Guide*

Establishing backup user equivalence

Before you use the **Backup Manager** to back up a filesystem on a remote host, you must establish user equivalence for the *backup* user account on both the remote and local hosts. To do this, use the **User Equivalence Manager**. On the local system:

1. Select **Open** from the **File** menu.
2. Enter or select the *backup* user and click on **OK**.
3. Select **Add Equivalence** from the **User** menu.
4. Enter the name of the remote host and enter **backup** in the "Remote user" field.
5. Select **Save** from the **File** menu.

Repeat steps 1 through 5 on the remote system.

Make sure that the */usr/backup/.rhosts* files on both the local and remote systems are owned by the *backup* user:

```
-rw-------    1 backup   backup        878 Oct 21 15:57 .rhosts
```

To verify that *backup* user equivalence is established, check the contents of the */usr/backup/.rhosts* files. For example, the *.rhosts* file on *pdxbox* contains:

```
hawthorne                backup
```

The *.rhosts* file on *hawthorne* contains:

```
pdxbox          backup
```

See also:

- "Adding user equivalence" in the *Networking Guide*

Examining the backup history

In the **Backup Manager** (page 106), select a filesystem, then select **Backup History** from the **Examine** menu. You see the "backup history", a list of all the backups that have been performed of the selected filesystem.

The backup history includes:

- the "Date" the backup was performed.
- the "Backup Type" (Level 0 (complete), Level 1, or Level 2).
- whether the "File List" was saved.
- the "Media Type" on which the filesystem was backed up.

If you have specified that the **Backup Manager** save the backup contents lists for a filesystem, the "File List" column shows "Yes". In this case, you can view the filenames contained in a particular backup by selecting the backup to browse and clicking on the **Browse File List** button. See "Browsing backup file lists" (page 128).

To save the backup contents lists when you run scheduled backups, select **Modify Filesystem Schedule** from the **Schedule** menu, click on the **Backup Options** button and select **Save Backup Contents Lists**.

About the backup history

When you perform a scheduled backup of a filesystem, the **Backup Manager** automatically records the following information about that backup:

- name of the filesystem

- names of the files and directories that were backed up (optional)
- date the scheduled backup was performed
- backup level that was used

This information is stored on your system in the backup history.

If your disk is small or full, you can specify not to save the names of the files and directories by modifying the backup schedule options (or the default backup schedule options) and deselecting **Save Backup Contents Lists**. See "Modifying scheduled filesystem backup options" (page 122).

To examine the backup history for a particular filesystem, see "Examining the backup history" (page 127).

When you restore a partial (files or directories) or complete filesystem from your scheduled backup volumes, the **Backup Manager** uses the backup contents lists to determine which backup volumes are required to restore the latest version of those files. The backup volumes required to restore files to a specific date are called the "backup set"; before starting a restore, you should retrieve this set of backup media from storage.

For example, if you are restoring a file that was not changed recently, the backup set might contain only the last complete (Level 0) backup.

Browsing backup file lists

In the **Backup Manager** (page 106), you can browse the file lists for scheduled backups or on media contents.

To look at files in a scheduled backup, do one of the following:

- Select the filesystem from the list, select **Backup History** from the **Examine** menu, select the backup to browse, and click on the **Browse File List** button.
- Select the filesystem from the list, select **From Backup History** from the **Restore** menu, select **Directories or Files**, and click on the **Select** button.

To look at files on media (not a scheduled backup), do one of the following:

- Select **Media Contents** from the **Examine** menu, then verify and insert the volume into the drive.
- Select **From Current Media** from the **Restore** menu, select **Files Selected From Media List**, click on **Select Files**, and insert the specified media volume.

While viewing the filenames in the list, you can display the files in a directory by double-clicking on the directory name. To return to the previous directory listing, click on the **Show Parent Directory** button.

To search for a specific filename on the media, click on **Search for Pattern**, enter the pattern, and click on **OK**. You can use these Bourne shell wildcard characters in the search pattern:

* match any string, including the null string

? match any single character

For example, to search for all files with "bin" in the filename, enter the following search pattern:

bin

See "Wildcard characters" in the *Operating System User's Guide* for more information.

Examining the contents of a backup

In the **Backup Manager** (page 106):

1. Select **Media Contents** from the **Examine** menu.
2. If necessary, select or enter the correct "Media Device" (page 109), then click on **OK**.
3. When prompted, insert each backup volume.
4. If the media contains a scheduled filesystem backup, the **Backup Manager** reads the file list from the first volume and displays the list of all the files contained in the backup.

   If the media contains an unscheduled backup or another backup not known to the current system, the **Backup Manager** reads all the volumes of the backup and displays a list of all the files contained in the backup.

   At this point, you can view the contents of directories or search for a pattern (page 128).

Restoring a scheduled filesystem backup

In the **Backup Manager** (page 106), to restore a scheduled filesystem backup:

1. Select the filesystem to restore.
2. Select **From Backup History** from the **Restore** menu.

> **NOTE** If you have not performed a scheduled backup of the filesystem, you see a message like:
>
> `Cannot get backup set for` ***system***

3. Specify the date to which to restore your filesystem in the "Restore to Date" field.

   The **Backup Manager** selects the backup set required to restore the filesystem based on this date. By default, the **Backup Manager** restores the filesystem to the current date. To restore your filesystem to a different date, enter the date in "Restore to Date".

4. To restore the filesystem to a destination other than the original location, click on the **Destination** button, select "Alternate Location", specify the "Destination Host" (if desired) and "Destination Directory", and click on **OK**.

   When you restore the filesystem to the original location, the **Backup Manager** extracts your files to the mounted filesystem, wherever it is currently mounted.

   The restoration might overwrite existing files. To be safe, you can restore into a temporary directory, such as */tmp*.

   NOTE The directory to which you restore a backup must be at least as large as the filesystem you are restoring.

5. Select **Complete Filesystem**.
6. When you are ready to start the restore, click on **Start**.
7. At the confirmation prompt, click on **OK** to start the restore. To view the list of files to be restored, click on **View Selected Files**.
8. Confirm or change the "Media Device" (page 109).
9. Insert the specified backup into the drive.
10. When prompted, insert each media volume.
11. When the restoration is complete, click on **OK**.

See also:

- "How backups restore complete filesystems" (this page)
- "Running scheduled backups" (page 112)

How backups restore complete filesystems

When you restore a complete filesystem, the **Backup Manager** constructs a backup set consisting of the last occurrence of each backup level, in ascending order.

For example, a hardware failure ruins the data on your hard disk on Friday, the last day (Day 5) of the backup schedule, immediately before the backup scheduled to be done that evening. The following backups were performed during the week:

1. Level 0 (Monday, Day 1)
2. Level 1 (Tuesday, Day 2)
3. Level 2 (Wednesday, Day 3)
4. Level 2 (Thursday, Day 4)

The **Backup Manager** constructs a backup set to restore the system to the current date (Day 5). The backup set consists of the Level 0, Level 1, and the Level 2 that was performed on Day 4. You do not need to restore the first Level 2 (Day 3), because the Level 2 that was performed on Day 4 backed up all the files changed since Day 2. The only information that is missing is what was changed during the day on Friday prior to the hardware failure.

See also:

- "Understanding incremental backups" (page 124)
- "Restoring a scheduled filesystem backup" (page 129)

Restoring files from a scheduled filesystem backup

In the **Backup Manager** (page 106):

1. Select the filesystem from which to restore files or directories.
2. Select **From Backup History** from the **Restore** menu.

   NOTE If you have not performed a scheduled backup for the filesystem, you see a message like:

   ```
   Cannot get backup set for system
   ```

3. Specify the date to which to restore your directories or files in the "Restore to Date" field.

   The **Backup Manager** selects the backup set required to restore the files based on this date. By default, the **Backup Manager** restores the files to the current date. To restore your files to a different date, enter the date in "Restore to Date".

4. To restore the files to a destination other than the original location, click on the **Destination** button, select **Alternate location**, specify the host and alternate directory, then click on **OK**.

The restoration might overwrite existing files. To be safe, restore into a temporary directory, such as */tmp*.

NOTE The directory to which you restore a backup must be at least as large as the files and directories you are restoring.

5. Select **Files or Directories** and click on **Select**.
6. Select the files to restore and click on **OK**. See "Selecting directories or files to restore" (page 133).
7. Click on **Start**.
8. At the confirmation prompt, click on **OK** to start the restore. To view the list of files to be restored, click on **View Selected Files**.
9. When prompted, insert each specified backup media volume into the drive and click on **OK**.
10. When the restoration is complete, click on **OK**.

See also:

- "How backups restore complete filesystems" (page 130)
- "Running scheduled backups" (page 112)

Restoring files or directories from backup media

In the **Backup Manager** (page 106), to restore the contents of a backup that is not a scheduled filesystem backup:

1. Select **From Current Media** from the **Restore** menu.
2. Select or enter the "Media Device" (page 109) from which to restore the backup.
3. Specify what is to be restored:
   - **Complete Contents of Media**
   - **Files Selected from Media List**

     This allows you to select from a list of files or directories on the media (which must be read first to generate the list). Click on **Select Files**, insert the media in the drive, then click on **OK**. Then, select the files to restore. See "Selecting directories or files to restore" (page 133).

- **Specified Files**

  This allows you to simply enter file or directory names without waiting for the manager to provide you with a list. You must enter the full pathname without the leading /, as in this example:

  well/world/gate

  To restore a directory along with its contents (instead of just the empty directory), as in this example:

  well/world/hex/*

  Click on **OK** when you are finished.

4. Specify the "Host" and "Destination directory" to which to restore the media.

   The restoration might overwrite existing files. To be safe, restore into a temporary directory, such as */tmp*.

   NOTE The directory to which you restore a backup must be at least as large as the files and directories you are restoring.

5. To start the restore, click on **Start**.
6. At the confirmation prompt, click on **OK** to start the restore. To view the list of files to be restored, click on **View Selected Files**.
7. When prompted, insert each media volume.
8. When the restoration is complete, click on **OK**.

See also:

- "Examining the contents of a backup" (page 129)
- **restore**(ADM) manual page (for command-line interface)
- "Selecting another host to manage" in the *Networking Guide*

Selecting directories or files to restore

To select directories or files to restore:

1. Select "Directories or Files" and click on the **Select** button.

   If you specified to save the backup contents lists (page 128), the "Backup File" list contains the filenames from the backup contents list. Continue with step 2.

   If you did not chose to save the backup contents lists, this information is not available online. Skip to step 3.

2. Select files or directories to restore from the "Backup File" list and click on the **Add** button. The files appear in the "Files to Restore" list.

   To display files in a directory, double-click on the directory. To return to the previous directory listing, click on **Show Parent Directory**.

   To search for a specific filename on the media, click on **Search for Pattern**, enter the pattern, and click on **OK**. You can use the "*" and "?" Bourne shell wildcard characters in the search pattern. See "Browsing backup file lists" (page 128) for more information.

   Instead of selecting from the "Backup File" list, you can enter specific files or directories to restore by clicking on the **Specify Files** button. When prompted, enter the file or directory name to restore and click on **Add** or press ⟨Enter⟩. The file appears in the "Files to Restore" list. Enter as many filenames as you like. When you finish selecting files to restore, click on **Close**.

   To remove files from the "Files to Restore" list, select the files and click on **Remove**. Remove all the files from this list by clicking on **Remove All**.

3. If the "Backup File" list is not available online, you can do one of the following:

   - Select files from the media list.

     When prompted, insert each specified backup media volume and click on **OK**. The **Backup Manager** reads the media and displays a list of files contained on the media in the "Backup File" list. Continue with step 2 above.

   - Specify filenames to restore.

     Enter the file or directory name and click on **Add** to add the filename to the "Files to Restore" list. You must enter the full pathname, but omit the leading /, as in this example:

     ```
     well/world/gate
     ```

     To restore the contents of a directory, use the "*" Bourne shell wildcard character, as in this example:

     ```
     well/world/hex/*
     ```

     When you finish specifying filenames, click on **OK**.

   When you finish selecting files to restore, click on **OK**.

See also:

- "Restoring a scheduled filesystem backup" (page 129)
- "Restoring files from a scheduled filesystem backup" (page 131)
- "Restoring files or directories from backup media" (page 132)

Specifying the Backup Manager default values

You can set these **Backup Manager** default values:

- Default backup schedule (page 122)
- Default backup device (this page)
- Default media options (page 136)

When you customize these **Backup Manager** defaults, you do not have to set these values each time you:

- create backups (scheduled or unscheduled).
- examine media contents.
- restore backups (scheduled or other backups).
- schedule filesystems (local or remote) to be backed up.

Setting the default backup device

In the **Backup Manager** (page 106), select **Defaults** from the **Options** menu, then select **Device**. Select the "Media Device" (page 109), then click on **OK**.

NOTE To add a device that is not listed, see "Setting the default media values" (page 136).

This sets the media device that appears whenever you:

- run a scheduled backup (page 112).
- run an unscheduled backup (page 119).
- examine the contents of a backup (page 129).
- restore a scheduled backup (page 129).
- restore files or directories from backup media (page 132).

This is useful, for example, if you want to run all of your scheduled backups on the cartridge tape drive.

See also:

- "About media devices" (page 109)
- "Setting the default media values" (page 136)

Setting the default media values

In the **Backup Manager** (page 106), set up options for each media device on your system or add a new device:

1. Select **Defaults** from the **Options** menu, then select **Media Values**.
2. For each device (page 109) on your system, specify:
   - "Block Size" (page 110)
   - "Volume Size" (page 110)

   You can add devices not supported by the Device Query Interface (page 109) (DQI) by entering the device name (for example, */dev/rct0*) and specifying the block and volume size.
3. Click on **Apply**.
4. Verify that the new values are correct and click on **OK**.
5. When you finish specifying options, click on **Close**.

Now, whenever you use a media device, such as Primary Floppy 1.2MB, the appropriate block size and volume size are used.

To remove entries for a non-DQI media device, select it and click on **Delete**. (The **Delete** button will not work on DQI devices.)

To specify a different default media device, see "Setting the default backup device" (page 135).

See also:

- "About media devices" (page 109)

Using the command line to create and restore backups

You can also perform backup and restore operations using the **cpio** utility on the command line. The examples included here assume you are using the standard tape device, */dev/rct0*. For more information, see **cpio**(C).

To create a backup:

find . -depth -mount -print | cpio -oHcrcB -C 10240 -K ***volumesize*** **-O /dev/rct**

where ***volumesize*** is the size of the media used. For example, use **120000** for a 120 MB tape drive. The **find**(C) command is used to create the list of files for the backup. The **-mount** option prevents backing up mounted filesystems.

NOTE Use of the **-K** option ensures that you are prompted to insert additional media when the current volume is filled. If you are using a large capacity storage device and changing volumes is unnecessary, you can omit the **-K** option.

To restore a backup:

cpio -iAmudB -I /dev/rct0

Restore individual files or directories by adding them the end of the command line. Remember to use the full pathname and omit the leading /, as in this example:

tmp/hold/time

Use the "*" Bourne shell wildcard character to restore a directory, as in this example:

tmp/hold/other/*

To verify a backup:

cpio -itvn -I /dev/rct0

To list the files on a backup:

cpio -iABmudq -I /dev/rct0 > *

This command reads the first file off the media. Backups created by the **Backup Manager** include a file list called *_BACKUP_CONTENTS_* at the beginning of each backup. If the backup was created by another means, use this command to list the files:

cpio -itv -I /dev/rct0

See also:

- **cpio**(C)
- "Creating a backup with cpio" in the *Operating System User's Guide*

Troubleshooting the Backup Manager

Most error messages displayed by the **Backup Manager** are self-explanatory. In other cases, the error message will be generic, as in these examples:

```
Could not get authorization data on host hostname
Backup of filesystem name failed
```

Error boxes of this type include a **Details** button to provide you with additional information. The problems reported fall into these categories:

- Remote administration problem (this page)
- Missing or corrupted database files (this page)

Remote administration problem

These are the most common error messages displayed under *Details* when encountering problems with remote administration:

`Could not retrieve the mount point for` ***host*** `/dev/`***filesystem***

`Failed to retrieve local filesystem list for host` ***host***
These messages occur when making an **Open Host** or **Schedule ⇨ New Remote Filesystem** selection without user equivalence for the account that is running the manager. When performing an operation that opens a remote host, the account on the local host must be recognized by the remote system. This differs from *backup* equivalency, which is required for remote administration (where you do not actually log in to the host using the **Open Host** dialog box).

`Error occurred while accessing the media device: Permission denied.`
This occurs when performing backing up a remote filesystem without user equivalence for the *backup* account on the remote host. See "Establishing backup user equivalence" (page 126) for more information.

`Could not get authorization data on host` ***hostname***
The remote system may be unreachable or there may be a configuration problem. See "Troubleshooting TCP/IP" in the *Networking Guide* for more information. This error is also diaplayed when performing a remote backup without user equivalence for the *backup* account on the remote host. See "Establishing backup user equivalence" (page 126) for more information.

Missing or corrupted database files

The **Details** may indicate file data (or the file itself) is missing, as in these instances:

`Failed to retrieve backup schedules`
One (or both) of the files */var/opt/K/SCO/Unix/*/sa/backup/hostname* or */var/opt/K/SCO/Unix/*/sa/backup/sched/**hostname*** are missing. Restore any missing files from backups.

`Could not access history files for` ***system*** `/dev/`***filesystem***
Certain files under */var/opt/K/SCO/Unix/*/sa/backup/history/**filesystem*** are missing. Restore the contents of this directory from backups.

Chapter 4

Managing printers and print jobs

Most systems require the ability to print out data on paper. SCO systems support a wide variety of printers. Use the **Printer Manager** (page 142) to:

- add local printers (page 144).
- add remote printers (page 146).
- change printer names or connections (page 154).
- specify the default printer (page 154).
- remove printers (page 151).
- enable and disable printers (page 152).
- accept or reject jobs (page 152).

See also:

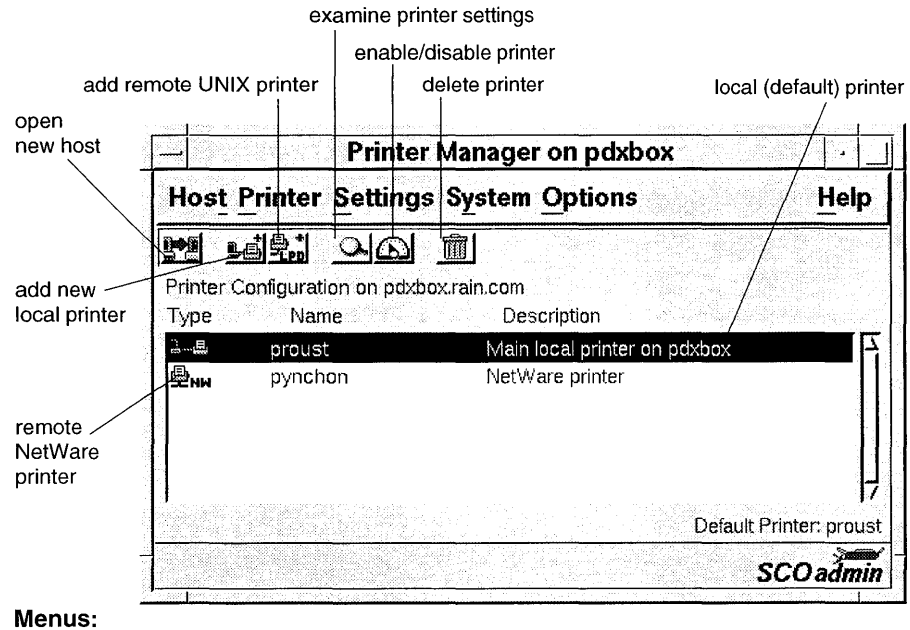
- "About the print service" (page 159)
- "Managing print jobs" (page 168)
- "Attaching a printer to a serial terminal" (page 206)
- "Controlling access to printers" (page 157)
- "Customizing printer configuration" (page 176) — including page size, banners, forms, filters, print wheels, font cartridges, alerts, spoolers, and interface scripts

The Printer Manager interface

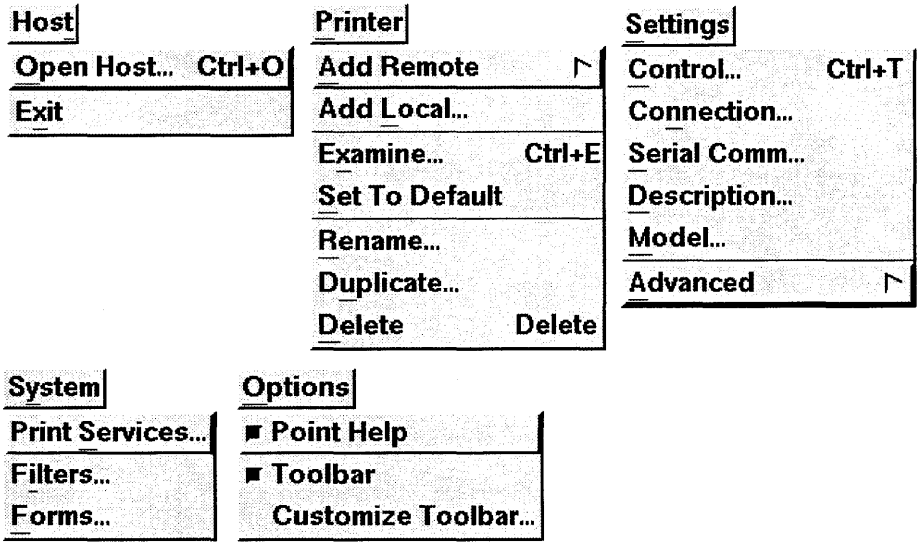
Use the **Printer Manager** to add and remove printers and manage the print service. You can start the **Printer Manager** in any of these ways:

- Double-click on the **Printer Manager** icon in the *Printers* directory of the *System Administration* window on the Desktop.
- Start the SCOadmin launcher by entering **scoadmin** on the command line, then selecting **Printers**, then selecting **Printer Manager**.
- Enter **scoadmin printer manager** on the command line (or abbreviate to **scoadmin printer**).

For more information on using SCOadmin managers, see "Administering your system with SCOadmin" in the *SCO OpenServer Handbook*.


examine printer settings
enable/disable printer
add remote UNIX printer
delete printer
local (default) printer
open new host
Printer Manager on pdxbox
Host Printer Settings System Options Help
add new local printer
Printer Configuration on pdxbox.rain.com
Type Name Description
proust Main local printer on pdxbox
pynchon NetWare printer
remote NetWare printer
Default Printer: proust
SCOadmin

Menus:


Host
Open Host... Ctrl+O
Exit
Printer
Add Remote
Add Local...
Examine... Ctrl+E
Set To Default
Rename...
Duplicate...
Delete Delete
Settings
Control... Ctrl+T
Connection...
Serial Comm...
Description...
Model...
Advanced
System
Print Services...
Filters...
Forms...
Options
Point Help
Toolbar
Customize Toolbar...

Adding local printers

A "local printer" is a printer attached directly to your system. After physically connecting your printer to your system (see Chapter 22, "Adding printers" in the *SCO OpenServer Handbook*), you must add the printer to the list of those that the print service recognizes.

In the **Printer Manager** (page 142):

1. Select **Add Local** from the **Printer** menu.

   NOTE You can do this remotely using the **Open Host** selection of the **Host** menu. This requires user equivalence on the machine you plan to administer, as described in "Adding user equivalence" in the *Networking Guide*. As on the local machine, non-*root* accounts require the **lp** authorization to run the **Printer Manager**. See "Assigning subsystem authorizations" (page 32) for more information.

   To add a new local printer that is similar to an existing local printer, simply copy the configuration from the existing printer and make small changes, instead of starting from scratch. See "Duplicating a local printer" (page 145).

2. Enter a unique name for the new printer.
3. Enter a description of the printer (optional).

   Use the "Description" field to identify the printer's location, brand name, or the group to which the printer belongs.

4. Select the printer model name that most closely matches your printer from the "Model" list.

   The default model (interface script) is **standard**. For information on customized printer scripts, see "Creating printer interface scripts" (page 182).

5. If the default device listed is incorrect, select the device to which the printer is connected from the "Device" list. If it does not appear in the list, enter the correct device in the "Device" field.

   The same device can be associated with more than one printer. See "About printer device connections" (page 155).

   NOTE If the device is a serial device that is not set to the default parameters, set the serial parameters. See "Changing printer names and connections" (page 154).

6. When you complete your choices, click on **OK**.

The new printer appears in the list in the **Printer Manager** window and is

enabled and configured to accept local jobs by default. Now, you can:

- configure the printer to accept remote jobs (page 152).
- change the printer settings (page 154).
- disable the printer (page 152).

See also:

- "Modifying printer creation defaults" (page 155)
- "Specifying the system default printer" (page 154)
- "Connecting to remote UNIX system printers" (page 146)
- "Removing local or remote printers" (page 151)
- "Setting up a print server" in the *Networking Guide*
- **lpadmin**(ADM) manual page (for command-line interface)
- **standard**(ADM) manual page (for command-line interface)

Duplicating a local printer

If you are adding a new local printer that is similar to an existing local printer, you can copy the configuration from an existing printer and make small changes, instead of starting from scratch.

In the **Printer Manager** (page 142):

1. Select the name of the printer to copy from the list of available printers.
2. Select **Duplicate** from the **Printer** menu.
3. Enter the name for the new printer.

The duplicated printer appears in the list in the **Printer Manager** window.

To customize the new printer, see "Changing printer names and connections" (page 154).

See also:

- "Adding local printers" (page 144)
- "Removing local or remote printers" (page 151)

Connecting to remote UNIX system printers

In the **Printer Manager** (page 142):

1. Select **Add Remote** from the **Printer** menu, then select **UNIX**.
2. Enter the name of the remote host or click on the **Select** button to choose from a list of remote hosts.
3. Enter the name of the remote printer (or printers) or click on the **Select** button to choose from a list of printers. If the remote system is not running SCOadmin (such as a non-SCO system or an older SCO system), you cannot use the **Select** button to select a printer; you must enter the printer name.
4. If the remote host is an SCO OpenServer system, you can select the **Use Extended Remote Printing Protocol** button. This allows you to move jobs between printers and enables the **lpstat**(C) command to display the status of remote printers.

   NOTE To access printer information on the remote host, you need user equivalence. This means the remote host must recognize the account being used to administer printers on the local machine. See "Adding user equivalence" in the *Networking Guide* for more information. As on the local machine, non-*root* accounts require the **lp** authorization to run the **Printer Manager**. See "Assigning subsystem authorizations" (page 32) for more information.

   When complete, the new remote printer appears in the list of printers available on the local host.
5. Make certain the remote host is prepared to accept jobs submitted from the local host. The local host must be listed in the in the */etc/hosts.lpd* or */etc/hosts.equiv* files on the remote host. The printer on the remote host must also be configured to accept remote jobs as described in "Accepting or rejecting print jobs" (page 152).

See also:

- "Removing local or remote printers" (page 151)
- "Adding local printers" (page 144)
- "Adding a SCO Gateway for NetWare printer" in the *Guide to Gateways for LAN Servers*
- "Configuring Hewlett-Packard network printers and print services" (page 147)
- "Managing shared printers" in the *SCO Advanced Server Administrator's Handbook*
- "Selecting another host to manage" in the *Networking Guide*

Configuring Hewlett-Packard network printers and print services

Hewlett-Packard network printers attach directly to a network using the built-in network card.

To configure a Hewlett-Packard network printer, use the **HP Network Printer Manager** located in the *Printers* directory of the SCOadmin hierarchy. To configure the print services for an HP network printer, use the **HP Network Print Services Manager** (in the same location).

There are three steps to configuring and managing HP network printers:

- Set up a BOOTP server (this page), the host which performs the actual configuration of the HP network printer using the **HP Network Print Services Manager**.
- Configure the HP print services on each host (page 148) that will access the printer using the **HP Network Print Services Manager**.
- Perform any further changes or additions (page 149) with the **HP Network Printer Manager**.

See the **hpnpcfg**(ADM) manual page for more information.

NOTE The **HP Network Print Services Manager** is used to initially install and configure the necessary files. The **HP Network Printer Manager** is used thereafter to configure printers and the print service. (The **HP Network Print Services Manager** actually invokes the **HP Network Printer Manager** for certain tasks, which is intentional.)

See also:

- **mkdev**(ADM) manual page (for command-line interface)
- **hpnpf**(ADM) manual page
- **bootpd**(ADMN) manual page
- **tftpd**(ADMN) manual page
- "Connecting to remote UNIX system printers" (page 146)

Setting up a BOOTP server

The BOOTP server must be on the same subnet as the HP network printer and no other host should be used to configure BOOTP for the printer. Follow these steps:

1. Start the **HP Network Print Services Manager**.

2. Enter **i** to install the network printing.
3. Enter **y** to install the startup configuration utilities.
4. Enter **y** to install the spooler utilities.
5. The **HP Network Printer Manager** is then invoked automatically. Enter **2** at the main menu to configure BOOTP/TFTP.
6. You are prompted to provide several pieces of information, including:
   - the MAC network card address of the printer (a 12 character string)
   - the network peripheral name
   - the IP address of the printer
   - whether to add the device to */etc/hosts*
   - the subnet mask (optional)
   - default gateway (optional)
   - syslog server (optional)
   - idle timeout
   - whether you want an access list
   - whether you want to configure SNMP parameters (including printer location, printer contact, "get" and "set" names, and SNMP traps)

   NOTE Do not use the **Verify BOOTP/TFTP configuration** option of the main menu; it is not supported.

7. Enter **6** to add the printer to the spooler.
8. Enter the name of the printer.
9. Enter the network name of the printer.
10. Enter the model interface script to use (generally *laserjethpnp*).
11. Enter the spooler class (optional).
12. Enter **y** if you want this to be the default printer.
13. Press ⟨Enter⟩ to continue and the is added to the system.
14. Reboot the system. The printer will be active upon startup.

Configuring hosts to use an HP network printer

On each host that will access the HP network printer:

1. Start the **HP Network Print Services Manager**.
2. Enter **i** to install the network printing.

3. Enter **n** to install the startup configuration utilities.
4. Enter **y** to install the spooler utilities.
5. Enter **6** to add the printer to the spooler.
6. Enter the name of the printer.
7. Enter the network name of the printer.
8. Enter the model interface script to use (generally *laserjethpnp*).
9. Enter the spooler class (optional).
10. Enter **y** if you want this to be the default printer.
11. Press ⟨Enter⟩ to continue and the printer is added to the system. It should be available for printing immediately.

Performing maintenance with the HP Network Printer Manager

Unless you need to remove the HP network print services entirely, use the **HP Network Printer Manager** to make any further changes once the service is configured.

Use the **HP Network Printer Manager** to:

- verify installation of software, printer connectivity, and printer operation
- configure a printer with BOOTP/TFTP
- add a printer to host
- remove a printer BOOTP/TFTP configuration
- remove a printer from service on a host

Configuring an UUCP dialup printer

A "UUCP dialup printer" is a printer that connects directly to a system through a dialup modem using UUCP to queue jobs on the remote system. For more information on UUCP, see Chapter 7, "Connecting to other computers with UUCP" (page 295).

To configure a dialup printer:

1. Set up your modem and verify that it works with **cu**(C). See "Configuring UUCP for modems" in the *SCO OpenServer Handbook.*

   Because **cu** accesses printers in the same way the print service does, set up the files as though preparing access to the printer for **cu**. The **cu** command is not used to access printers; however, if **cu** can access a printer, the print service can access it, too.

CAUTION The entry you use for the dialup configuration in the */usr/lib/uucp/Devices* file should use a *Dialers* entry rather than a dialer binary.

2. Add the following line to */etc/default/lpd*:

```
DIALUPPRINTER=YES
```

3. Enter the following:

   /usr/lib/lpadmin -p *printer_name* -m dumb -U *phone_number* -h -A mail

   This command sets up ***printer_name*** as the dialup printer; ***phone_number*** is the phone number for the remote system.

4. Set up the printer to accept jobs, and then enable the printer. See "Accepting or rejecting print jobs" (page 152) and "Enabling and disabling printers" (page 152).

5. Test the printer by submitting a job:

   lp -d *printer_name filename*

If the printer or port is busy, the print service automatically tries again. If the printer is busy, the retry rate is once every 10 minutes; if the port is busy, the retry rate is once every 20 minutes. The retry rates are not adjustable; however, you can force an immediate retry by enabling the printer. If the port or printer is likely to be busy for an extended period, disable the printer (page 152).

If an attempt to reach the dialup printer fails, **lpstat -p** reports the reason for a failed dial attempt. If you have set up fault alerting, the fault alert message reports the reason for the fault, see "Setting up printer fault alerts" (page 200). These messages are identical to the error messages produced by the UUCP system for similar problems, see "UUCP STATUS error messages" (page 347).

See also:

- **cu**(C) manual page
- **lp**(C) manual page
- **lpadmin**(ADM) manual page
- **lpstat**(C) manual page

Removing local or remote printers

In the **Printer Manager** (page 142), select the printer to remove, then select **Delete** from the **Printer** menu.

The printer is removed from the list in the **Printer Manager** window.

See also:

- "Adding local printers" (page 144)
- "Connecting to remote UNIX system printers" (page 146)
- **lpadmin**(ADM) manual page (for command-line interface)

Servicing printers and print services

Once you add your printer to the print service, the printer and print service are ready to accept jobs.

Occasionally, you may need to service the printer. For example, you might need to perform maintenance on or fix the printer, or change the printer configuration.

NOTE You can perform maintenance remotely using the **Open Host** selection of the **Host** menu. This requires user equivalence on the machine you plan to administer, as described in "Adding user equivalence" in the *Networking Guide*. As on the local machine, non-*root* accounts require the **lp** authorization to run the **Printer Manager**. See "Assigning subsystem authorizations" (page 32) for more information.

Here are some tasks you might need to perform periodically:

- "Starting and stopping the print services" (page 153)
- "Enabling and disabling printers" (page 152)
- "Accepting or rejecting print jobs" (page 152)
- "Mounting and unmounting forms" (page 191)
- "Changing a font cartridge on a printer" (page 199)
- "Troubleshooting the print system" (page 215)

Enabling and disabling printers

In the **Printer Manager** (page 142), select **Control** from the **Settings** menu, then select **Enable printing.**

When you add a printer with the **Printer Manager**, the printer is enabled automatically. When you want to perform maintenance or halt printing for another reason, disable the printer. A disabled printer continues to accept print jobs (the jobs are added to the queue), but does not print them. If the printer was printing a job when it was disabled, the printer will start the print job from the beginning when it is enabled.

Enabling printing allows the scheduler, **lpsched**, to print files on the printer.

See also:

- "Managing print jobs" (page 168)
- "Accepting or rejecting print jobs" (this page)
- "Starting and stopping the print services" (page 153)
- **enable**(C) manual page (for command-line interface)
- **disable**(C) manual page (for command-line interface)
- **lpsched**(ADM) manual page (for command-line interface)

Accepting or rejecting print jobs

In the **Printer Manager** (page 142), select **Control** from the **Settings** menu, then select **Accept new remote jobs** or **Accept new local jobs.**

Normally, when you add a printer with the **Printer Manager**, the printer is set up to accept local print jobs automatically. When a printer is accepting print jobs, **lp** adds new jobs to the printer's queue.

When you specify that a printer reject print jobs, all new print requests are denied (not added to the queue) until you set the printer to accept them again. If a user tries to print a file on a printer that is not accepting requests, the print job is not accepted and **lp** displays this message:

```
UX:lp: ERROR: Requests for destination "printer_name" aren't
              being accepted.
```

NOTE If a printer is enabled and is set to deny requests, it continues to print jobs already in the queue. To delete these jobs, see "Deleting print jobs" (page 170).

See also:

- "Managing print jobs" (page 168)
- "Enabling and disabling printers" (page 152)
- "Starting and stopping the print services" (this page)
- **accept**(ADM) manual page (for command-line interface)

Starting and stopping the print services

The print services start automatically each time the system goes into multi-user mode and stops when you bring down the system. Under normal circumstances, you should never have to start or stop the print service manually. (For example, you do not have to stop the print service to change printer configurations or to add forms or filters.) However, you can stop the print service manually without stopping the operating system.

In the **Printer Manager** (page 142), select **Print Services** from the **System** menu, then select:

- **Local print service enabled**

  (If NetWare is installed, the option is **Local/NetWare print service enabled**)

- **Remote UNIX print service enabled**

Stopping the print service causes all printing to cease within seconds.

When you start the print service, the printer configurations, forms, and filters that were in effect when you stopped the print service are restored. It might take a minute or two for these printer configurations to be reestablished before any saved print requests start printing. Any print requests that did not finish printing when the scheduler stopped are printed in their entirety when the print service starts.

NOTE Jobs can appear to pass through a printer that is not online. If a printer is not online or operating properly, you should disable the printer (page 152).

See also:

- **lpsched**(ADM) manual page (for command-line interface)

Changing printer names and connections

After adding a new local or remote printer with the **Printer Manager**, you can **check printer settings** by selecting the printer from the list, then selecting **Examine** from the **Printer** menu.

To change the name of a local printer, select the printer from the list, select **Rename** from the **Printer** menu, then enter the new name. The new name appears in the **Printer Manager** window in the "Name" field.

To change the description, model, connection (device), or serial communication parameters for a printer, select the printer from the list, select the appropriate item from the **Settings** menu, then select the setting from the displayed list of options.

Select the printer model (interface script) that most closely matches your printer. For information on customized printer scripts, see "Creating printer interface scripts" (page 182).

NOTE You cannot change the name or model of a remote printer. When you select **Connection** from the **Settings** menu for a remote printer, you see the host to which the remote printer is attached.

See also:

- "About printer device connections" (page 155)
- "About serial communication parameters" (page 156)
- "Connecting to remote UNIX system printers" (page 146)
- "Specifying the system default printer" (this page)
- **lpadmin**(ADM) manual page (for command-line interface)

Specifying the system default printer

The system default printer is the destination that is used when a user does not set the default Desktop printer (with **Print Setup** from the **File** menu of the application), does not explicitly specify a destination (from the command line), or does not set the **LPDEST** shell variable.

In the **Printer Manager** (page 142), select the printer from the list and select **Set To Default** from the **Printer** menu.

The message at the bottom right of the window indicates that this printer is now the system default printer.

To set the system default destination to a class, enter at the command line:

/usr/lib/lpadmin -d ***class_name***

See "About printer classes" (page 158).

To set the default printer for a user, set the **LPDEST** shell variable in that user's *.profile* or *.cshrc* file.

- For Bourne or Korn shells (*.profile*):

  LPDEST= ***printer*****;export LPDEST**

- For C shell (*.cshrc*):

  setenv LPDEST ***printer***

See also:

- **lpadmin**(ADM) manual page (for command-line interface)

Modifying printer creation defaults

In the **Printer Manager** (page 142), select **Defaults** from the **Options** menu.

You can change the defaults used for the creation of new printers or the addition of remote printers:

Device default device file (this page)

Model default interface script (page 180)

Terminfo Type default terminfo type (page 185)

Enable printing
enable printing by default (page 152)

Accept new local/remote jobs
accept print jobs by default (page 152)

For more information about adding local and remote printers, see "Adding local printers" (page 144) and "Connecting to remote UNIX system printers" (page 146).

About printer device connections

Each local printer must have a "device special file" associated with it. Device special files provide an interface to a device (such as a printer) and its corresponding device driver. These files reside in the */dev* directory.

NOTE Remote UNIX and NetWare printers are associated with a remote host instead of a device.

The default local printer connections are three parallel devices (*/dev/lp0, /dev/lp1, /dev/lp2*) and two serial devices (*/dev/tty1a* for COM1 and */dev/tty2a* for COM2).

To use these devices, you might need to add them to the system using **mkdev parallel** or **mkdev serial**. You can also use these **mkdev** commands to determine which devices are currently configured on your system. See the **mkdev**(ADM) manual page and Chapter 20, "Adding serial and parallel ports" in the *SCO OpenServer Handbook*.

NOTE When specifying a serial device, make certain that you use the non-modem control device (for example: */dev/tty3a*, not */dev/tty3A*).

See also:

- "About serial communication parameters" (this page)

About serial communication parameters

Each serial device is controlled by a series of communication characteristics that define the low-level communications with the printer. When you configure a new printer, the default values should be sufficient for most purposes.

The serial communication parameters are:

Baud Rate Speed of the serial connection. The default speed is 9600bps; other choices are 300, 1200, 1800, 2400, 4800, 19200, and 38400.

Parity Type of parity. The default parity is "none"; other choices are "odd" and "even".

Flow Control XON/XOFF flow control (handshaking) is enabled by default; other choices are "none" and "hardware".

Flow control is used between the computer and the device to start and stop the flow of data when necessary (as when the device input buffer is full).

Character Size Number of bits (7 or 8) per character. Character size is set to 8 bits by default (not to be confused with font size, which is the size of a printed character in points).

To change these parameters, see "Changing printer names and connections" (page 154).

You can configure additional serial line characteristics using **lpadmin**(ADM) or by editing the print interface file (in the */usr/spool/lp/model* directory) directly. See "Creating printer interface scripts" (page 182).

The **stty**(C) manual page summarizes the various characteristics that can be set on a terminal or printer port; however, not all of the characteristics listed in **stty**(C) are important for printers.

Controlling access to printers

You can limit the use of a printer to a subset of all users on your computer. For example, you might want to restrict access to a printer if it is reserved for printing sensitive information and only a subset of the people can print sensitive information or if using a high-quality printer incurs expenses that not all people are allowed to incur.

To restrict use of the printer, the print service uses the lists of users allowed or denied for a printer. If a user tries to use a restricted printer, the print service checks the list and refuses the request if the user is not allowed to use the printer.

If you do not add user names to the allow or deny lists, the print service assumes that everybody can use the printer.

In the **Printer Manager** (page 142), select **Advanced** from the **Settings** menu, then select **Users**.

- **To deny access to the printer**, select the users from the "Allowed" list and click on **Deny**.
- **To allow access to the printer**, select the users from the "Denied" list and click on **Allow**.

See also:

- **lpadmin**(ADM) manual page (for command-line interface)

About printer classes

If you have a large number of printers, you might find it convenient to group a collection of printers together into a "printer class". Print requests sent to a class of printers are printed by the first available member of that class. This allows faster turnaround, as printers are kept as busy as possible. In addition, users do not need to know which printer is idle. (Users can still send print jobs to a specific printer using the **-d** option to **lp**(C) from the command line.)

One way to use classes is to group a series of printers that should be used in a particular order into a class. For example, you have a high-speed printer and a low-speed printer and you want to direct as many print requests as possible to the high-speed printer and use the low-speed printer only when the high-speed printer is busy. The print service always checks for an available printer in the order that the printers were added to a class. If you add the high-speed printer to the class before adding the low-speed printer, the print service routes print requests in the order you want. See "Grouping printers into a class" (this page).

NOTE You do not need to set up classes if you only want to allow users to submit print requests to a type of printer. Users can specify the printer class with **lp**(C):

lp -T *content_type*

The first available printer that can handle that type of file prints the file. See "About content types" (page 194).

Grouping printers into a class

To group printers into a printer class, add each printer to the class using the following command:

/usr/lib/lpadmin -p *printer_name* **-c** *class_name*

For example, to add printer *snapper* to the class *fish*, enter:

/usr/lib/lpadmin -p snapper -c fish

If the class does not exist, **lpadmin** creates it.

NOTE Class names and printer names must be unique. This allows a user to specify the destination of a print request without having to know whether it is a class of printers or a single printer. Thus, you cannot have a class and printer with the same name.

A printer does not belong to any class until you add it to one.

To list all the classes and the printers that belong to them, enter:

lpstat -c

You see a display similar to:

```
members of class fish:
        snapper
        salmon
```

To remove a printer from a class, enter:

/usr/lib/lpadmin -p *printer_name* -r *class_name*

For example, remove printer *salmon* from class *fish*:

/usr/lib/lpadmin -p salmon -r fish

If the printer was the only member of the class, the class is removed.

To remove a class, enter:

/usr/lib/lpadmin -x *class_name*

Removing a class does not remove the printers that were members of the class.

NOTE You can remove a printer or class only if it has no pending print requests. If there are pending requests, you must first transfer them to another printer or class or remove them. See "Transferring a job to another printer" (page 172) and "Deleting print jobs" (page 170).

If the printer or class removed is also the system default destination, the system no longer has a default destination.

See also:

- "About printer classes" (page 158)
- **lpadmin**(ADM) manual page
- **lpstat**(C) manual page

About the print service

The UNIX system print service is a collection of utilities that help you, as system administrator (or **lp** administrator), to configure, monitor, and control the printers on your system. See Figure 4-1, "Overview of print request processing" (page 161).

The print service:

- receives files users want to print.
- 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 mounting forms and filters.
- issues error messages when problems arise.

When a user sends a file to a printer, the print service assigns the request (or "print job") a unique name, the "request ID". The request ID consists of the name of the printer on which the file is to be printed and a unique number identifying the file. Use this request ID to find out the status of the print job or to cancel the print job. See "Deleting print jobs" (page 170) and "Viewing jobs in the print queue" (page 170). The print service keeps track of all the print requests in the request log (page 162).

The print job is "spooled", or lined up, with other print jobs to be sent to a printer. Each print job is processed and waits its turn in line to be printed. This line of pending print jobs is called a "print queue". Each printer has its own queue; you can hold jobs in the queue (page 171), move jobs up in a queue (page 172), or transfer jobs to another queue (page 172).

Overview of print request processing

The diagram in Figure 4-1, "Overview of print request processing" (page 161), gives an overview of the processing of a print request.

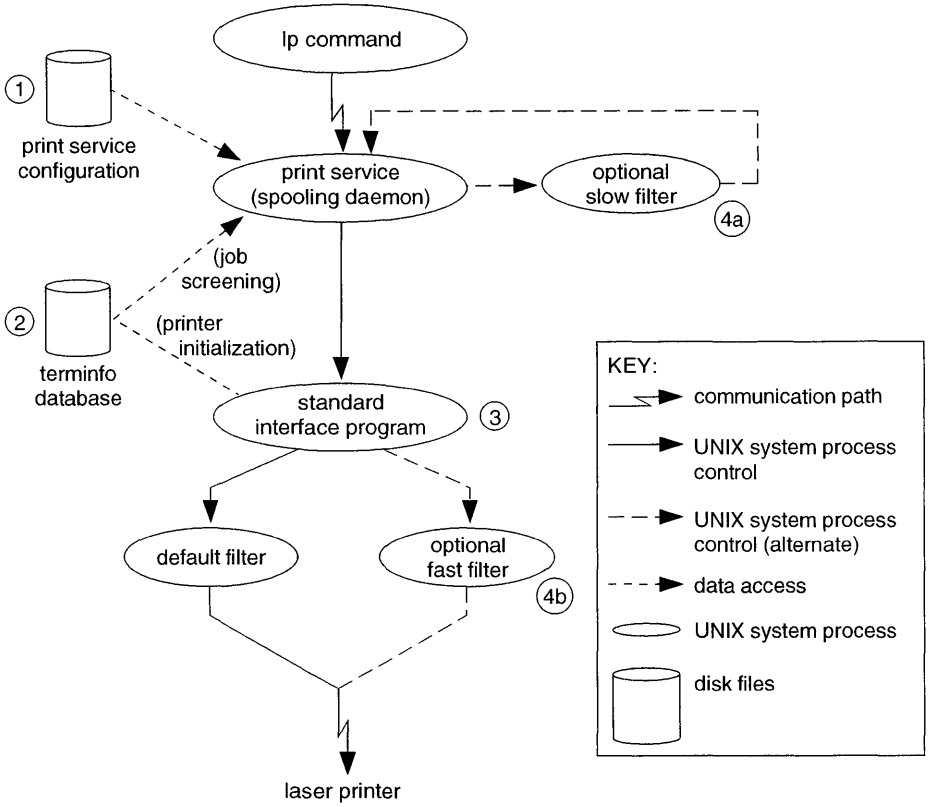


Figure 4-1 Overview of print request processing

Each print request is sent to a "spooling daemon" (background program) that keeps track of all the jobs. (This information is archived in the request log.) The daemon is created when you start the print service (page 153). The spooling daemon is also responsible for keeping track of the status of the printers and slow filters; when a printer finishes printing a job, the daemon starts it printing another job if one is queued.

You can customize the print service by adjusting or replacing some of the items shown in Figure 4-1 (this page) (the numbers are keyed to the diagram).

1. For most printers, you need only change the printer configuration stored on disk. "Changing printer names and connections" (page 154) explains how to do this. Some of the more printer-dependent configuration data are the printer port characteristics (page 156).
2. For printers that are not represented in the *terminfo* database, you can add a new entry that describes the capabilities of the printer. See "Adding a printer entry to the terminfo database" (page 186). The print

service uses the *terminfo* database 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 requests.

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 interface program uses the same information to initialize the printer.

3. If you have a particularly complicated printer or if you want to use features not provided by the print service, you can change the interface script. This script 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, for example, mapping one set of escape sequences into another, and can provide a special setup by interpreting print modes requested by a user. Slow filters are run separately by the spooling 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.

See also:

- "Managing print jobs" (page 168)
- "Customizing printer configuration" (page 176)
- "About the print request log" (this page)

About the print request log

Each time a user sends a job to the printer, the print service creates two files that describe the job request and places one each in the */usr/spool/lp/temp* and */usr/spool/lp/requests* directories. The information about the job is split into two files so that the system can keep sensitive information secure in the */usr/spool/lp/requests* directory. The user who submitted the job has access to the request file in */usr/spool/lp/temp*; only the **lp** administrator (or *root*) has access to the file in */usr/spool/lp/requests*.

The request files remain in these directories only while the job is in the queue. When the job finishes printing, the information in the two files is combined and appended to the "request log", */usr/spool/lp/logs/requests*.

The structure of the request log is simple, which makes it easy to extract data using common UNIX shell commands. The requests are listed in the order in which they were printed, separated by lines that begin with the request ID.

Each line below the separator line is marked with a single letter, the "request log code", that identifies the kind of information contained in the line. Each letter is separated from the data by a single space. Table 4-1, "Request log entries" (this page), describes these codes. Here is a sample entry from the print request log:

```
= ps-717, uid 1532, gid 18, size 7872, Tue May 10 14:43:10 1994
z ps
C 1
D ps
F /usr/spool/lp/temp/717-1
P 20
t simple
U hanna
s 0x0010
```

Table 4-1 Request log entries

Letter	Content of line
=	The separator line lists the (comma-separated) request ID, user ID (**uid**) and group ID (**gid**) of the user who submitted the request, total number of bytes in the original (unfiltered) file (**size**), and the date and time the request was queued.
C	Number of copies printed.
D	Printer or class destination or the word "any".
F	Name of the file in the */usr/spool/lp/temp* directory. This line is repeated for each file printed, and files are printed in the order given.
f	Form name used (if applicable).
H	Type of special handling used: **resume**, **hold**, or **immediate**.
N	How the print service notified the user after printing the file (if applicable): **M** by an electronic mail message **W** by a message written to the user's terminal
O	Any **-o** options given to **lp**(C).
P	Priority of the print request (page 174), if applicable.
p	List of pages printed.

(Continued on next page)

Table 4-1 Request log entries
(Continued)

Letter	Content of line
r	Any **-r** options given to **lp**(C) indicating that the user requested raw processing of the file.
S	Character set or print wheel used.
s	Outcome of the job, expressed as a combination of individual bits in hexadecimal form. The important bits used internally by the spooler are: **0x0004** Slow filtering finished successfully. **0x0010** Printing finished successfully. **0x0040** Request was canceled. **0x0100** Request failed filtering or printing.
T	Title on the banner page.
t	Content type of the file.
U	Name of the user who submitted the print request.
x	Slow filter.
Y	List of special modes to give to the filters used to print the request.
y	Fast filter.
z	Printer used for the request. This differs from the destination (the **D** line) if the request was queued for "any" printer or a class of printers, or if the **lp** administrator transferred the request to another printer.

See also:

- "Overview of print request processing" (page 160)
- **lp**(C) manual page (for command-line interface)
- **lpstat**(C) manual page (for command-line interface)

Print service command summary

In general, you should use the **Printer Manager** and **Print Job Manager** to manage your print service. However, if you need to manage your print service from the command line, refer to these commands:

- User-level commands (this page)
- **printerstat** authorization commands (this page)
- **lp** authorization commands (page 167)

Table 4-2 Print service commands available to all users

Command	Description	See also
cancel(C)	Cancels a request for a file to be printed	"Deleting print jobs" (page 170)
lp(C)	Sends a file or files to a printer	
lpstat(C)	Reports the status of the print service	"Viewing jobs in the print queue" (page 170)
lprint(C)	Prints from printer attached to a terminal	

The administrator can give users the ability to disable and enable a printer so that if a printer is malfunctioning, the user can turn the printer off without having to call the administrator. (However, in your printing environment, it might not be reasonable to allow regular users to disable a printer.)

You can control whether other users have access to the commands in Table 4-3, "Privileged print service commands" (this page), by assigning or revoking the **printerstat** authorization. See "Secondary authorizations" (page 34) for more information about **printerstat**.

Table 4-3 Privileged print service commands

Command	Description	See also
disable(C)	Deactivates the named printer(s)	"Enabling and disabling printers" (page 152)
enable(C)	Activates the named printer(s)	"Enabling and disabling printers" (page 152)

Table 4-4, "Administrative print service commands" (page 167) lists print service commands available only to the administrator. To use the administrative commands, you must either be logged in as *root* or have the **lp** authorization. See "Assigning subsystem authorizations" (page 32) for information about setting the **lp** subsystem authorization.

NOTE Only *root* can run **lpsched** and **lpshut**.

The administrative print service commands are located in the */usr/lib* directory. If you use these commands frequently, include */usr/lib* in your **PATH** variable.

Table 4-4 Administrative print service commands

Command	Description	See also
accept(ADM)	Permits jobs to be queued for a specified destination	"Accepting or rejecting print jobs" (page 152)
reject(ADM)	Prevents jobs from being queued for a specified destination	"Accepting or rejecting print jobs" (page 152)
lpadmin(ADM)	Sets up or changes printer configurations	"Changing printer names and connections" (page 154)
lpfilter(ADM)	Sets up or changes filter definitions	"Creating and using printer filters" (page 191)
lpforms(ADM)	Sets up or changes pre-printed forms (use **/usr/lib/lpadmin** to mount a form)	"Creating and using printer forms" (page 188)
lpmove(ADM)	Moves output requests from one destination to another	"Transferring a job to another printer" (page 172)
lpsched(ADM)	Starts the print service	"Starting and stopping the print services" (page 153)
lpshut(ADM)	Stops the print service	"Starting and stopping the print services" (page 153)
lpusers(ADM)	Sets or changes the default priority and priority limits the users of the print service can request	"Setting print queue priorities" (page 173)

Managing print jobs

Use the **Print Job Manager** (this page) to manage print jobs on all the printers available to the system.

All users can delete, hold, and resume their own jobs. If you have **lp** authorization (or are logged in as *root*), you can also promote jobs to the top of the print queue for a particular printer, transfer jobs to another printer, and manipulate any job (not just your own) in the list.

See also:

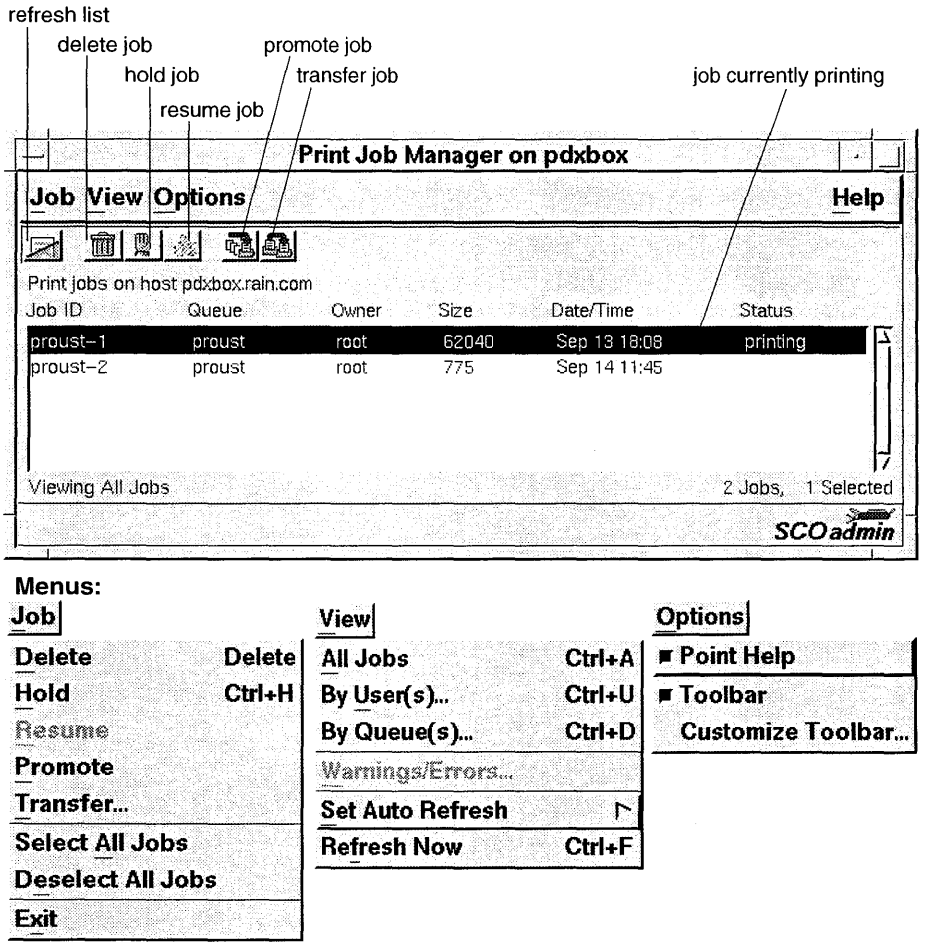
- "Selecting and deselecting multiple jobs" (page 169)
- "Viewing jobs in the print queue" (page 170)
- "Deleting print jobs" (page 170)
- "Holding and resuming print jobs" (page 171)
- "Transferring a job to another printer" (page 172)
- "Moving jobs to the top of the queue" (page 172)

The Print Job Manager interface

Use the **Print Job Manager** to delete, hold, resume, promote, and transfer print jobs. You can start the **Print Job Manager** in any of these ways:

- Double-click on the **Print Job Manager** icon in the *Printers* directory of the *System Administration* window on the Desktop.
- Start the SCOadmin launcher by entering **scoadmin** on the command line, selecting **Printers**, then selecting **Print Job Manager**.
- Enter **scoadmin print job manager** on the command line (or abbreviate to **scoadmin p**).

For more information on using SCOadmin managers, see "Administering your system with SCOadmin" in the *SCO OpenServer Handbook*.



See also:

- "Customizing the toolbar" (page 213)

Selecting and deselecting multiple jobs

Two options are available to make selection easier in the **Print Job Manager**:

- To select all jobs at once, select **Select All Jobs** from the **Job** menu.
- To deselect all jobs, select **Deselect All Jobs** from the **Job** menu.

Viewing jobs in the print queue

A "print queue" is a list of all the print jobs waiting to print on a particular printer. Because you can manage jobs on all the printers available to your system, the **Print Job Manager** displays the jobs in multiple queues in one list by default.

To change the listing of print jobs, use the the **View** menu:

- To view only the jobs that belong to particular users, select **By User(s)**. Select the user(s) from the "Available" list and click on **Add**. The users appear in the "Selected" list.

  To remove users from the "Selected" list, select the user and click on **Remove**. The users are removed from the "Selected" list.

- To view only the jobs queued on particular printers (or in printer classes), select **By Queue(s)**. Select the queue(s) from the "Available" list and click on **Add**. The queues appear in the "Selected" list.

  To remove queues from the "Selected" list, select the queues and click on **Remove**. The queues are removed from the "Selected" list.

- To view all the jobs in all the queues, select **All Jobs**.

The status of print jobs change frequently (particularly on a busy system) and the list might not reflect those changes. To update the list, see "Refreshing data in the display" in the *SCO OpenServer Handbook*.

See also:

- "Selecting and deselecting multiple jobs" (page 169)
- **lpstat**(C) manual page (for command-line interface)

Deleting print jobs

In the **Print Job Manager** (page 168), to delete (cancel) a job, select the job (or jobs) to delete, then select **Delete** from the **Job** menu (or click on the **Delete** button). Click on **OK** to confirm.

The "Status" column changes to `canceled` and then the job disappears from the display.

NOTE If you are not allowed to delete a job (if, for example, the job belongs to another user), the **Delete** menu item and button are inactive (dimmed). Unless you are logged in as *root* or you have **lp** authorization, you cannot delete jobs belonging to other users. See "Assigning subsystem authorizations" (page 32) for information about **lp** authorization.

See also:

- "Selecting and deselecting multiple jobs" (page 169)
- "Viewing jobs in the print queue" (page 170)
- **cancel**(C) manual page (for command-line interface)

Holding and resuming print jobs

In the **Print Job Manager** (page 168), select the job (or jobs) to hold, then select **Hold** from the **Job** menu (or click on the **Hold** button).

The "Status" column changes to `held` to indicate that the job will not be printed until you resume it. You can hold any job that has not already finished printing.

NOTE If you are not allowed to hold a job (if, for example, the job belongs to another user or the job is already being held), the **Hold** menu item and button are inactive (dimmed). Unless you are logged in as *root* or you have **lp** authorization, you cannot hold and resume jobs belonging to other users. See "Assigning subsystem authorizations" (page 32) for information about **lp** authorization.

To resume a held print job, select the job to resume, then select **Resume** from the **Job** menu (or click the **Resume** button). The "Status" column changes to indicate that the job is no longer being held. The **Resume** menu item and button are inactive (dimmed) unless you select a job that is being held.

If the job was printing when you held it, it becomes the next job to print when you resume it. Normally the job starts printing from the beginning at page one. To specify that the held job resume printing at a later page, enter the following at the UNIX command line:

lp -i ***request-id*** **-H resume -P** ***starting-page-***

The ***request-id*** is the code for the print job in the "Job ID" column on the main screen. The final dash (**-**) is required to print both the starting page and all subsequent pages.

The ability to print a subset of pages requires a filter that supports this feature. The default filter used by the print service does not support this. If the appropriate filter is not installed on the printer, any attempt to resume a print job on a later page will be rejected. See "Creating and using printer filters" (page 191).

See also:

- "Selecting and deselecting multiple jobs" (page 169)
- "Viewing jobs in the print queue" (page 170)
- **lp**(C) manual page (for command-line interface)

Transferring a job to another printer

You cannot transfer jobs unless you have **lp** authorization (or are logged in as *root*). See "Assigning subsystem authorizations" (page 32) for information about **lp** authorizations.

In the **Print Job Manager** (page 168), select the job (or jobs) to transfer, then select **Transfer** from the **Job** menu (or click on the **Transfer** button). Then, select the printer (or printer class) to which the job is to be moved and click on **OK**. Click on **OK** to confirm the transfer to the new printer. The job will now be printed on the new printer.

See also:

- "Viewing jobs in the print queue" (page 170)
- **lpmove**(ADM) manual page (for command-line interface)

Moving jobs to the top of the queue

You cannot move jobs up in the queue unless you have **lp** authorization (or are logged in as *root*). See "Assigning subsystem authorizations" (page 32) for information about **lp** authorizations.

By default, the print service adds jobs to the queue for a particular printer in the order they are requested: the first job submitted appears at the top of the queue and thus is printed first. However, if you want a job to be printed sooner, you can promote it (move it up the queue).

In the **Print Job Manager** (page 168), select the job, then select **Promote** from the **Job** menu (or click on the **Promote** button). The job moves up the queue.

Promoting a job does not necessarily mean that the job will print first:

- If there is a job currently printing, the job moves to next in line in the queue.
- If other jobs have already been moved up, the job moves to the end of the list of promoted jobs.

To print a critical job ahead of any others, preempting the currently printing job, first hold the currently printing job (click on the **Hold** button), then promote the critical job (click on **Promote**). This delays the current print job until you **Resume** it and allows the urgent print job to print immediately. See "Holding and resuming print jobs" (page 171).

See also:

- "Viewing jobs in the print queue" (page 170)
- "Setting the priority level for a print job" (page 174)
- "Setting print queue priorities" (this page)

Setting print queue priorities

The print service provides a simple priority mechanism that allows users to adjust the position of a print job in the queue. When users submit print jobs with **lp**(C), they can assign a "priority level" of between 0 and 39 to the print jobs. A *lower* number indicates a *higher* priority. Requests with higher priority are placed ahead of print jobs with lower priority. See "Setting the priority level for a print job" (page 174).

To ensure that the priority scheme works, *root* or the **lp** administrator controls how high users can set the priority by:

- **assigning each user a priority limit** (page 175).
  Users cannot submit print jobs with a priority higher than their individual limit (users can submit jobs with lower priority levels).
- **assigning a default priority limit for the users not assigned a personal limit** (page 175).
  This limits the priority that users who do not have a personal priority limit can set.
- **setting a default priority level for the print service** (page 175).
  This priority is assigned to print jobs 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).

To list the current default priority level and priority limits, enter:

/usr/lib/lpusers -l

If you have not set any priority limits or defaults, you see:

```
Default priority: 20
Priority limit for users not listed below: 0
Priority  Users
```

If you set the priority limit for *steve* and *naomi* to 10, when you list the priority level and limits, you see:

```
Default priority: 20
Priority limit for users not listed below: 0
Priority  Users
   10     steve,naomi
```

See also:

- **lpusers**(ADM) manual page

Setting the priority level for a print job

Users can set a priority level for a specific print job using the **-q** option with the **lp**(C) command from the UNIX command line. The priority level values range from 0 (highest priority) to 39 (lowest).

For example, to print the file *urgent* and set the priority level to 5 (relatively high priority), enter:

lp -q 5 urgent

If the user does not specify a priority level for a print job, the print service uses the default priority level; initially, this is set to 20. To change the default priority level, see "Setting the default priority level" (page 175).

The **lp** administrator (or *root*) can change the priority for any print job that is already in the queue, as long as it is not already printing:

lp -i *request-id* -q *new-priority-level*

This repositions the print job in the queue, putting it ahead of lower priority jobs, 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 the administrator overrides this limit.

See also:

- "Setting individual and default priority limits" (page 175)
- "Setting print queue priorities" (page 173)
- **lp**(C) manual page

Setting individual and default priority limits

The "priority limit" is the highest priority level a user can assign to a job submitted to the print service with **lp**(C). See "Setting the priority level for a print job" (page 174).

To set the priority limit for a particular user, enter:

/usr/lib/lpusers -q ***priority_level*** **-u** ***username***

The ***priority_level*** is a number between 0 (highest) and 39 (lowest).

To set the priority limit for a group of users, list their names after the **-u** option, separated by commas or spaces (enclose the list in quotes if you use a space).

For example, to set the priority limit for *steve* and *naomi* to 10, enter:

/usr/lib/lpusers -q 10 -u "steve naomi"

The "default priority limit" is the limit for all users who have not been assigned a personal limit.

To set the default priority limit, enter:

/usr/lib/lpusers -d ***priority_level***

For example, if you set the default priority limit to 30 and then list the priority limit values with **/usr/lib/lpusers -l**, you see:

```
Default priority: 20
Priority limit for users not listed below: 30
Priority  Users
   10     steve,naomi
```

To remove the individual priority limit for a user (and return the priority limit to the default for that user), enter:

/usr/lib/lpusers -u ***username***

If you do not set a default priority limit, users without personal limits are limited to priorities between 20 and 39.

See also:

- "Setting print queue priorities" (page 173)
- **lpusers**(ADM) manual page

Setting the default priority level

When a user sends a file to the printer without specifying a priority level, the print service assigns the system-wide default priority level (initially 20) to the print job.

To change the default priority level, enter:

/usr/lib/lpusers -d ***priority_level***

The ***priority_level*** is a value between 0 (highest priority) and 39 (lowest priority).

For example, if you change the default priority level to 30, then list the priority information with **/usr/lib/lpusers -l**, you see:

```
Default priority: 30
Priority limit for users not listed below: 0
Priority  Users
```

Do not confuse the default priority level with the "default priority limit"; the default priority limit prevents users who have not been assigned a personal limit from requesting too high a priority. See "Setting individual and default priority limits" (page 175).

NOTE If the default priority is greater than the priority limit for a user, the limit is used instead.

See also:

- "Setting print queue priorities" (page 173)
- **lpusers**(ADM)

Customizing printer configuration

Although the print service is flexible enough to handle most printers and printing needs, you might require features that the standard print service does not accommodate.

You can customize the print service by:

- "Setting default printer page size and spacing" (page 177)
- "Bypassing the spooler" (page 178)
- "Specifying the number of banners" (page 179)
- "Creating printer interface scripts" (page 182)
- "Adding a new printer manually" (page 185)
- "Creating and using printer forms" (page 188)
- "Creating and using printer filters" (page 191)
- "Specifying character sets" (page 197)

- "Specifying font cartridges to use with a printer" (page 199)
- "Setting up printer fault alerts" (page 200)

See also:

- "Setting up a printer with multiple names" (page 204)
- "Attaching a printer to a serial terminal" (page 206)
- "Configuring a spooled local terminal printer" (page 209)
- "Initializing parallel printers with an init device file" (page 212)

Setting default printer page size and spacing

When a user submits a request to print a file, the page size, character pitch, and line pitch (spacing) are normally determined from the form that it is printed on. If the user does not require a form, he or she can specify the page size and print spacing to use. If the user gives neither a form to use nor the page size and print spacing, defaults are used.

NOTE This information applies only to dot-matrix and print wheel printers, not to PostScript printers.

By setting defaults for each printer, you can make submitting print requests easier. For example, you can designate different printers as having different default page sizes or print spacing; you can dedicate one printer to printing wide (132-column) output, another to printing normal (80-columns, 66 lines) output, and yet to another printing letters in monospaced fonts (12 characters per inch, 8 lines per inch). Users simply route their file to the appropriate printer to get the style of output they want.

You can specify four default settings: page width, page length, character pitch, and line pitch. Specify the first two in columns and lines, inches (i) or centimeters (c); the last two in characters per inch (cpi) or lines per inch (lpi).

In addition, specify the character pitch as **pica** for 10 cpi, **elite** for 12 cpi, or **compressed** for the maximum cpi the printer can provide (up to a limit of 30 cpi).

To specify the default settings, use the following commands:

/usr/lib/lpadmin -p ***printer_name*** **-o width=***scaled-number*
/usr/lib/lpadmin -p ***printer_name*** **-o length=***scaled-number*
/usr/lib/lpadmin -p ***printer_name*** **-o cpi=***scaled-number*
/usr/lib/lpadmin -p ***printer_name*** **-o lpi=***scaled-number*

NOTE **lpadmin**(ADM) uses the printer type to determine whether the settings are possible for the printer. Therefore, you must first set the printer type before you can specify these defaults.

For example, to specify a page width of 11 inches, a page length of 14 inches, character pitch to compressed, and line pitch of 3 lines per inch for the printer barney, enter:

```
/usr/lib/lpadmin -p barney -o width=11i
/usr/lib/lpadmin -p barney -o length=14i
/usr/lib/lpadmin -p barney -o cpi=compressed
/usr/lib/lpadmin -p barney -o lpi=3
```

If you do not provide defaults, the page size and print spacing are set to 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 **lpstat**(C) to display the printer configuration. To display the default page size and print spacing, enter:

lpstat -p ***printer_name*** **-l**

You see a display that includes information similar to:

```
Default pitch: compressed CPI 3 LPI
Default page size: Default page size: 11i wide 14i long
```

If you do not set the defaults, **lpstat** reports defaults from the *terminfo* database entry for the printer.

See also:

- **lpadmin**(ADM) manual page
- **lpstat**(C) manual page

Bypassing the spooler

If you use a printer without the spooler, any **stty**(C) settings you have specified for use with that printer do not remain 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:

1. Log in as *root*.

2. Insert one of the following lines in the initialization file */etc/rc2.d/S80lp* before the line that calls */usr/lib/lpsched* (the first command is for serial printers; the second for parallel printers):

   (stty *baud* ixon ixoff -ixany ; cat > /dev/null) < /dev/tty*n* &

   (stty onlcr; while : ; do sleep 3600; done) < /dev/lp*n* &

   baud is the baud rate of the printer, tty*n* and lp*n* are the serial and parallel device names, respectively.

   This command sets the **stty** options and holds the port open for use without the spooler. If you ever need to enable the port, you must kill the **stty** process first.

   NOTE With certain multiport cards, you must add a **sleep** command after the initialization program supplied with the card, followed by the **stty holdopen** command:

   ***initprogram* &**
   sleep 3

   initprogram is the program supplied with your multiport card.

See also:

- "Managing print jobs" (page 168)
- "About the print service" (page 159)

Specifying the number of banners

A "banner" is a page describing the print request (printer name, user, date, and so forth) that prints out with the print job. Normally, one banner prints with each request. However, you can choose not to print a banner page or you can specify the number of banner pages to print.

In the **Printer Manager** (page 142):

1. Select **Advanced** from the **Settings** menu, then select **Banners**.
2. Enter the number of banners to print (**0** for no banner), then click on **OK**.

See also:

- **lpadmin**(ADM) manual page (for command-line interface)

About printer interface scripts

A "printer interface script" is a program that the print service uses to manage the printer each time it prints a file. The interface script initializes the printer, takes advantage of its particular capabilities, prints the file, and reports any errors.

The printer interface scripts are associated with the printer model and are located in */usr/spool/lp/model*. For example, the printer interface script for a PostScript® printer is called */usr/spool/lp/model/postscript*. You can also create your own interface scripts or customize existing ones to suit your needs. See "Creating printer interface scripts" (page 182).

Interface scripts:

- initialize the printer port (the connection between the computer and the printer). The **standard** interface script uses **stty**(C) to initialize the printer port.
- initialize the physical printer (restore the printer to a normal state in case a previously printed file has left it in an unusual state), setting the character pitch, line pitch, page size, and character set requested by the user. The **standard** interface script uses **lp.set**(ADM) to initialize the printer.
- print banner page (or pages), if required.
- print the requested files. The **standard** interface script calls the **lp.cat**(ADM) printer filter to print the files.
- report any errors to the print service. The **standard** interface script uses **lp.tell**(ADM) to send descriptions of printer faults to the print service; the print service forwards that information as an alert to the print administrator.

The interface script does not open the printer port; the print service opens the printer port (or calls a dialup printer if that is how the printer is connected). The print service gives the printer port connection to the interface script as standard output and sets the printer to be the controlling terminal for the interface script. If the port experiences a hangup, a **SIGHUP** signal is sent to the interface script.

Many of the interface scripts provide special options that the user can specify with the **-o** option to **lp**(C). (Read the appropriate interface file for this information.) For example, the PostScript interface script includes the options listed in Table 4-5 (this page).

Table 4-5 PostScript options

Option	Description
land	prints text in landscape mode
land2	prints text in two-page landscape mode
port	prints text in portrait mode
raw	prints a PostScript file*

* Use the raw option for PostScript files only; the **PostScript** interface script converts text files to PostScript automatically if you do not specify raw.

The print service runs the interface script to send the print job to the printer:

/usr/spool/lp/admins/lp/interfaces/*printer id user title copies options file1 file2 ...*

Arguments to the interface script are:

printer The name of the interface script (the same as the 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 the user.

options List of blank-separated options, specified by the user (using **lp -o**) or by the print service (from default values specified by the administrator with **lpadmin**). See **lp**(C) for the list of options recognized by the **standard** interface.

file Full pathname of a file to be printed.

When the interface script is invoked, standard input comes from */dev/null*, standard output is directed to the printer port, and standard error output is directed to a file that will be displayed to the user who submitted the print request.

The print service passes additional printer configuration information to the interface script as shell variables:

TERM=*printer-type*
Specifies the printer type. The value is used as a key for getting printer capability information from the extended *terminfo* database.

FILTER='*pipeline*'
Specifies the filter to use to send the request content to the printer; the filter is given control of the printer.

CHARSET=*character-set*

Specifies the character set to use when printing the content of a print request. The **standard** interface script extracts the control sequences needed to select the character set from the *terminfo* database.

See also:

- "Creating printer interface scripts" (this page)
- **standard**(ADM) manual page

Creating printer interface scripts

If you have a printer that is not supported by simply adding an entry to the *terminfo* database, or if your printing needs are not supported by the **standard** or other interface scripts provided in */usr/spool/lp/model*, you can create your own printer interface script.

To create a customized interface script:

1. Start with the **standard** interface script (or one of the other scripts in */usr/spool/lp/model*) and modify it, rather than starting from scratch. For example:

   cd /usr/spool/lp/model
   cp standard okidatanew

2. Make sure that the custom interface script sets the proper **stty** modes (terminal characteristics such as baud rate or output options). Look for the section that begins with this line:

   ```
   ## Initialize the printer port
   ```

   Modify the code in the **standard** interface script. It sets both the default modes and the adjusted modes given by the print service or the user with a line like:

   stty *mode options* 0<&1

   This command line takes the standard input for the **stty** command from the printer port. For example, this **stty** command line sets the baud rate to 1200bps and sets some of the option modes:

   stty -parenb -parodd 1200 cs8 cread clocal ixon 0<&1

3. Set the hardware flow control printer port characteristic. The **standard** interface script does not set hardware flow control. This is set according to your computer hardware. The code for the **standard** interface script suggests where to set this and other printer port characteristics. Look for the section that begins with this line:

   ```
   # Here you may want to add other port initialization code.
   ```

4. Because different printers have different numbers of columns, make sure the header and trailer for your interface script correspond to your printer. The **standard** interface script 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 script that begins with this line:

   ```
   ## Print the banner page
   ```

5. Specify that the custom interface script print all user-related error messages to the standard output or to the standard error. The print service prints standard output errors on the page and mails standard error to the user.

6. Specify that when printing is complete, the interface script exits with a code that tells the status of the print job. Table 4-6, "Exit codes" (page 184), describes how the print service interprets exit codes.

   One way of alerting the administrator to a printer fault is to exit with a code of 129. Unfortunately, if the interface script exits, the print service reprints the print job from the beginning once the fault is cleared. To get an alert to the administrator without reprinting the entire job, specify that the interface script send a fault message to the print service, but wait for the fault to clear. When the fault clears, the interface script resumes printing the job. When finished printing, the interface script can exit with zero as if the fault never occurred. An added advantage is that the interface script can detect when the fault is cleared automatically so that the administrator does not have to reenable the printer.

   To specify that fault messages be sent to the print service, use **lp.tell**(ADM). The **standard** printer interface code calls **lp.tell**(ADM) with the **LPTELL** shell variable. The **lp.tell** program sends its standard input to the print service. The print service forwards the message as an alert to the administrator. If its standard input is empty, **lp.tell** does not initiate an alert. Examine the code immediately following these comments in the **standard** interface script for an example of how to use the **lp.tell** (**LPTELL**) program:

   ```
   # 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 **lp.tell**, there is no longer the need for the interface script to disable the printer itself. Your interface script can disable the printer directly, but doing so overrides the fault-alerting mechanism. Alerts are sent only if the print service detects that the printer has faulted and the special exit code and **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 script with a signal 15 (see the **signal**(S) and **kill**(C) manual pages for more information).

If the interface script dies from receipt of any other signal, the print service assumes that future print jobs are not affected and continues to use the printer. The print service notifies the person who submitted the print job that the job did not finish successfully.

The signals **SIGHUP**, **SIGINT**, **SIGQUI**, and **SIGPIP** (trap numbers 1, 2, 3, and 13) start out being ignored when the interface is invoked. The **standard** interface script changes this to trap these signals at appropriate times, interprets these signals to mean that the printer has a problem, and issues a fault.

Table 4-6 Exit codes

Code	Meaning to the print service
0	The print job completed successfully. If a printer fault occurred, it was cleared.
1 to 127	The print service encountered a problem in printing the job (for example, there were too many nonprintable characters or the job exceeded the printer's capabilities). This problem does not affect future print jobs. The print service should notify the person who submitted the print job (via **write**(C) or **mail**(C)) that an error occurred in printing the job. If a printer fault occurred, it was cleared.
128	Reserved for internal use by the print service. Interface scripts must not exit with this code.
129	The print service encountered a printer fault in printing the job. This problem affects future print jobs. If the fault recovery for the printer directs the print service to wait for the administrator to fix the problem, the print service should disable the printer. If the fault recovery is to continue printing, the print service should not disable the printer, but try printing again in a few minutes.
> 129	Reserved for internal use by the print service. Interface scripts must not exit with codes in this range.

See also:

- "Setting up printer interface scripts" (page 185)
- "About printer interface scripts" (page 180)

Setting up printer interface scripts

By default, the print service uses the standard interface script, */usr/spool/lp/model/standard*. This interface script should handle most of your printing needs. If you plan to use the standard interface program, simply select the **standard** model when you add the printer.

If the standard interface script does not meet your needs, you might be able to use one of the printer interface scripts provided in the */usr/spool/lp/model* directory. When you add the printer, select the model name from the list that most closely matches your printer. See "Adding local printers" (page 144). (You do not need to perform any special tasks to use either the standard interface or one of the other interfaces provided.)

To change the interface script after you add the printer, in the **Printer Manager**, select **Model** from the **Settings** menu.

If neither the standard interface script nor the scripts listed by the **Printer Manager** (or in */usr/spool/lp/model* directory) work for your printer, you can modify an existing printer interface script or create your own. See "Creating printer interface scripts" (page 182).

If you use a customized printer interface script, you must associate it with your printer. To do this, place the script in the */usr/spool/lp/model* directory. In the **Printer Manager** (page 142), select **Model** from the **Settings** menu and select the customized printer interface script from the list. When you associate a printer with a customized interface script, the **Printer Manager** copies the specified printer interface program to the */usr/spool/lp/admins/lp/interfaces* directory and renames the file ***printer_name***.

See also:

- "About printer interface scripts" (page 180)
- **lpadmin**(ADM) manual page (for command-line interface)
- **standard**(ADM) manual page

Adding a new printer manually

The print service relies on the standard interface script 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 (*/usr/lib/terminfo/terminfo.lp*) to add a new printer to the print service.

The *terminfo* database identifies each printer by a short name, identical to the kind of name used to set the **TERM** shell variable. For example, the name in the *terminfo* database for the AT&T model 455 printer is "455".

To specify the *terminfo* type for your printer, in the **Printer Manager** (page 142) select **Advanced** from the **Settings** menu, then select **Terminfo Type.** By default, the *terminfo* database includes entries for several popular printers. Select the *terminfo* type from the list that corresponds to your printer.

If *terminfo* does not include an entry for your printer, you might still be able to use the printer with the print service. However, you will not be able to use automatic selection of page size, pitch, and character sets, and you might have trouble keeping the printer set to the correct modes for each print request or using printer forms with the printer. In this case, you can either add an entry to *terminfo* (this page) for your printer or create a customized interface program (page 182) to use with the printer.

See also:

- **terminfo**(M) manual page

Adding a printer entry to the terminfo database

To create a *terminfo* entry for your printer:

1. Identify an entry in the */usr/lib/terminfo/terminfo.lp* file that uses the same or similar commands as the printer you are adding and copy that information to ***filename***, where ***filename*** is the file containing the *terminfo* entry you created for the printer.
2. Use the information in the manual for your printer, Table 4-7 (page 187), and the **terminfo**(M) manual page to modify the entry in ***filename***.
3. Once you create the new entry, compile it into the database:

   tic ***filename***

After adding or deleting *terminfo* entries or changing values that govern pitch settings, page width and length, or character sets, stop and restart the print service (page 153).

Table 4-7 terminfo entry definitions for printers

terminfo entry	Description
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
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	absolute vertical position
scs	select character set
smgb	set bottom margin at current line

(Continued on next page)

Table 4-7 terminfo entry definitions for printers
(Continued)

terminfo entry	Description
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

See **terminfo**(M) for the *terminfo* file structure and for information on how to construct a *terminfo* database entry for a new printer.

See also:

- "Adding a new printer manually" (page 185)
- **tic**(C) manual page
- *termcap & terminfo*, by John Strang, Linda Mui, and Tim O'Reilly, published by O'Reilly & Associates, Inc.

Creating and using printer forms

A preprinted "printer form" is a blank paper form that you load into your printer. An application typically generates a file that, when printed on the blank form, fills out the form. The print service includes facilities to create and administer forms.

To create a new printer form:

1. Create a form description file that specifies the format of the form.

For example, create a file */tmp/check.desc* and include all or any subset of the following information (substitute your information for the values in **bold**):

```
Page length: 66
Page width:  80
Number of pages: 2
Line pitch: 10
Character pitch: 16
Character set choice: any
Ribbon color: blue
Comment:
      Check form
Alignment pattern:

     XXXX XXXXXXXXXXXX XXXXXXXXX
                                  xxxxxx
     xxxxxxxxxxxxxxxxxxxxxxxxxx
```

The print service uses the alignment pattern to line up the forms before printing begins and prompts you to perform an alignment before printing.

Depending on your printer, specify page length in lines, inches (i), or centimeters (c). Specify page width in columns, inches (i), or centimeters (c). In the example above, page length is specified as **66** lines. If the printer recognizes inches, specify the page length as **11i**.

2. In the **Printer Manager** (page 142), select **Forms** from the **System** menu.
3. Click on **New**.

   Enter the name of the new form in the "Form Name" field and the name of the form description file in the "Filename" field.

   In our example, the filename is */tmp/check_desc*, and the name of the form is *check*. Choose a name that describes the form, because you use this name when you mount the form.

   Clicking on **OK** in the **New Form** window creates the directory (called *check*) for the form in */usr/spool/lp/admins/lp/forms* and the files necessary for the spooling subsystem to recognize the form. One of the files that is created in the form directory, *describe*, contains a copy of the form description file.

NOTE If you make changes to the form description file, you must modify the form.

4. To specify which users are allowed or denied access to the form, click on the **Users** button in the **System Forms** window:
   - **To deny access to the form,** select the user from the "Allowed" list and click on **Deny**.
   - **To allow access to the form,** select the user from the "Denied" list and click on **Allow**.
5. Specify which printers have access to the form by selecting the printer from the list, then selecting **Advanced** from the **Settings** menu, **Forms**, then **Allow**.
6. Mount the form. See "Mounting and unmounting forms" (page 191).

The form is now ready to use.

To specify the form with **lp**(C) from the command line, enter:

lp -f *form_name file_name* **-d** *printer_name*

When you specify a form to **lp**, the print system checks the printer for a mounted form. If no form is mounted, the job waits in the queue until you mount the appropriate form.

If you mount a form on a printer and then print a file that does not require the form to that printer, you must unmount the form. See "Mounting and unmounting forms" (page 191).

To examine an existing form, select **Forms** from the **System** menu. Then, select the form to examine and click on **Examine**.

To modify an existing form, select **Forms** from the **System** menu. Then, select the form to modify and click on **Modify**. Enter the name of the modified form description file in the "Filename" field. This associates the new form description file with the form name.

To remove a form from the system, select **Forms** from the **System** menu. Then, select the form to delete and click on **Delete**.

See also:

- "Mounting and unmounting forms" (page 191)
- "Alerting to mount forms and font cartridges" (page 203)
- **lpforms**(ADM) manual page (for command-line interface)
- **lpadmin**(ADM) manual page (for command-line interface)
- **lp**(C) manual page

Mounting and unmounting forms

Before the print service prints files that require a preprinted form, you must mount the form on a printer.

If you have set up alerting for the form, the print service alerts you when enough print requests are queued for it to be mounted. See "Alerting to mount forms and font cartridges" (page 203).

When you mount a form, you should verify that it is lined up properly. If you specified an alignment pattern with the form, you can request that this be printed repeatedly until you have adjusted the printer so that the alignment pattern looks correct. See "Creating and using printer forms" (page 188).

Mounting a form involves first loading the form onto the printer and then telling the print service that it is ready to mount. 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 should disable the printer first.

To mount the form:

1. Disable the printer (page 152).
2. Load the new form onto the printer.
3. In the **Printer Manager** (page 142), select the printer on which you want to mount the form from the list.
4. Select **Advanced** from the **Settings** menu, select **Forms**, then select **Mount**.
5. Select the form to mount from the list, then click on **OK**.
6. Reenable the printer (page 152).

To unmount a form, in the **Printer Manager** (page 142), select the printer on which the form is mounted. Select **Advanced** from the **Settings** menu, select **Forms**, then select **Mount**. Select "none" from the "Mounted forms" list.

NOTE Unless a form is mounted on a particular printer, the print service does not send files that require a form to that printer.

Creating and using printer filters

A "printer filter":

- converts a user's file into a data stream that prints properly on a given printer.
- handles the various modes of printing that users request, such as two-sided printing, landscape printing, draft or letter quality printing, and so on.

- detects printer faults and informs the print service, which in turn alerts you.

The print service supports forms and filters. However, there are very few cases where you need to use these features.

You can use filters to change the output to the printer. For example, you can create a filter that allows users to print a manual page by piping the output of **man**(C) to **lp**(C) without printing the control characters.

To create a filter:

1. Create a printer filter definition file.

   For example, create a file */tmp/printmandef* and include this information (substitute your information for the values in **bold**):

   ```
   Input types: nroff,man
   Output types: simple
   Printer types: any
   Printers: any
   Filter type: slow
   Command: col -b
   Options:
   ```

   Table 4-8, "Printer filter definition fields" (page 194), describes each of these fields.

2. Change the permissions on the filter definition file:

   chmod +x printmandef

3. In the **Printer Manager** (page 142), select **Filters** from the **System** menu.
4. Click on **New**.

   Enter the name of the new filter in the "Filter Name" field and the name of the filter definition file in the "Filename" field.

   In our example, the filename is */tmp/printmandef*, and the filter name is the name that you want the print service to assign to the form, for example, *printman*.

   Clicking on **OK** in the **New Filter** window adds the filter definition (*printman*) in the */usr/spool/lp/admins/lp/filter.table* file.

The filter is now ready to use.

To use the new **printman** filter to print the **lpfilter**(ADM) manual page without the control characters, enter:

man lpfilter | lp -d *printer_name* -Tman

Here is a sample filter definition that formats an ASCII document in PostScript so that a PostScript printer can read it:

```
Input types:
Output types: ps,postscript
Printer types: laser
Printers: hplaser
Filter type: fast
Command: /usr/spool/lp/bin/text2post
Options: MODES portrait = port, MODES landscape = land, MODES landscape2 = land2
```

To add the *printps* filter, proceed from step 3 (page 192). To use the *printps* filter to print a file in landscape mode with **lp**(C) on the printer *apple*, enter:

lp -d apple -Tps -y land *file*

Here is a filter definition that allows the user to define which pages to print:

```
Input types: simple
Output types: simple
Filter type: fast
Command: /usr/spool/lp/bin/pages
Options: PAGES * = *
```

To add the *printpage* filter, proceed from step 3 (page 192). To use the *printpage* filter to print specific pages, enter:

lp -d laser -P 3-5,9 *file*

In this case, **lp** prints only pages 3 through 5 and page 9 of *file* on the printer *laser*.

Here is a filter definition that translates a PostScript file to epson format for printing on a dot matrix printer:

```
Input types: postscript ps
Output types: epson
Filter type: slow
Commands: /usr/spool/lp/bin/ps2epson
Options: MODES draft = -d, MODES low = -d
```

To add the *printepson* filter, proceed with step 3 (page 192) of the procedure above. To use the *printepson* filter to print a PostScript file on the Epson dot matrix printer, enter:

lp -d dotepson -Tps *file*

To examine an existing filter, select **Filters** from the **System** menu. Then, select the filter to examine and click on **Examine**.

To modify an existing filter, select **Filters** from the **System** menu. Then, select the filter to modify and click on **Modify**. Enter the name of the modified filter definition file in the "Filename" field. This associates the new filter definition file with the filter name.

To remove the filter from the system, select **Filters** from the **System** menu. Then, select the filter to delete and click on **Delete.**

Table 4-8 Printer filter definition fields

Field	Description
Input types	List of file types that the filter can process.
Output types	List of file types that the filter can produce as output. This should match the content type of at least one printer.
Printer types	List of printer types into which the filter can convert files. For most printers and filters, you can use the default ("any") or leave this field blank.
Printers	List of printers that can use this filter. For most filters, you can use the default ("any") or leave this field blank.
Filter type	Type of filter (either fast or slow). A fast filter is one that incurs little overhead in preparing a file or one that must have direct access to the printer. A slow filter is the opposite. The print service runs slow filters in the background, allowing files that need fast filtering (or no filtering) to be printed first.
Command	Full pathname of the command that defines the filter program. Include any fixed options that the filter always requires.
Options	Options to **lp**(C). These **lp** options are converted into options for the filter.

See also:

- **lpfilter**(ADM) manual page (for command-line interface)
- **lp**(C) manual page

About content types

The "content type" tells the print service what types of files can be printed on a particular printer type (or interface script). 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, however, can accept several different types of file and print their contents properly. When adding this kind of printer, specify the names of the content types the printer accepts.

When a user submits a file 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.

To specify the content types that a printer can accept:

1. In the **Printer Manager** (page 142), select the printer from the list.
2. Select **Advanced** from the **Settings** menu, then select **Content Types**.
   - **To add a defined content type** to the list of those supported by the printer, select the type from the "Defined" list, then click on **Add**.
   - **To remove a content type** from the list of those supported by the printer, select the type from the "Supported" list, then click on **Remove**.
   - **To define a new content type**, click on **Define New**, enter the new content type, and click on **OK**.

     Content type names can contain no more than 14 characters and can include only letters, digits, and underscores. Choose content type names that mean something to you and the users of the printer.

If several different types of printer accept the same content type, use the same content type names when you add those printers. This makes it easier for users because they can use the same name to identify the type of file they want to print, regardless of the printing destination.

For example, several manufacturers produce printers that accept PostScript files. Although these printers might require different interface scripts so each can be properly initialized, they all might be capable of printing the same content type, which you might call *PostScript*. As another example, several manufacturers produce printers that accept ANSI X3.64-defined escape sequences. However, the printers might not support all the ANSI capabilities, or each might support different sets of capabilities. You can specify different content type names for these printers to differentiate them.

The print service recognizes special meanings for these content types:

simple The most common type of file on the UNIX system is known as *simple*. The print service assumes this file type contains only printable ASCII characters and the backspace, tab, line feed, form feed, and carriage return control characters. See the **lpadmin**(ADM) manual page for more information.

terminfo The *terminfo* content type name does not refer to a particular type of file, but instead refers to all the types represented in the *terminfo* database (page 185). It is not likely that any printer is capable of handling all the types listed in the database, but this name is reserved for describing possible filter capabilities.

any The print service reserves the content type name *any* for describing the types of files a filter can accept or produce. This name should not be used as a content type when adding a printer.

If a printer can handle *simple* file types, include *simple* in the list of "Supported" content types. If you do not want a printer to accept *simple* file types, you must include an alternate list of content types that the printer accepts. (If no other content type is appropriate, use the printer interface script.)

If you do not define the types of files a printer can accept, the print service assumes the printer can print only files of content type *simple*. This may be sufficient if users choose the proper printer and make sure the files are properly prepared for the printer before they are submitted for printing.

Detecting printer fault indicators with filters

Just as converting a file and handling special printing modes is printer-specific, so is printer fault detection. The print service attempts to do this in general, and for most printers, it properly detects faults. However, it is limited to checking for "hangups" and excessive delays in printing. The print service cannot determine the cause of the fault, so it cannot tell you exactly what to look for.

A properly designed filter can provide better fault coverage. Some printers can send a message to the host describing the reason for a fault. Others indicate a fault by dropping the carrier or shutting off data flow. A filter can serve you by giving more information about a fault and detecting more of them.

Filters can wait for a printer fault to clear and then resume printing. This allows for more efficient printing when a fault occurs because the print job that was interrupted does not have to be reprinted in its entirety. Only a filter that understands the control sequences used by a printer knows where a file breaks into pages. Thus, only the filter knows how far back to go in the file to restart properly.

The print service has a simple interface that lets the filter get the fault information to you and restart the print job if it can. The alerting mechanism is handled by the print service. The interface program that manages the filter takes all error messages from the filter and places them into an alert message that can be sent to you. Thus, you see any fault descriptions that the filter puts out. If you set the printer configuration so that printing should automatically resume after a fault is cleared, the interface program keeps the filter active so that it can pick right up where it left off.

See also:

- "Creating and using printer filters" (page 191)
- "Setting up printer fault alerts" (page 200)
- **lpfilter**(ADM) manual page

Font cartridges, character sets, print wheels

Printers differ in the way they print different font styles. Some have font cartridges, some have changeable print wheels, others have preprogrammed, selectable character sets. (Print wheels are used on older, so-called "daisy wheel" impact printers that use small wheels with the print characters around the perimeter.) The print service can minimize the impact of these differences on the users of the print service.

You can specify which font cartridge, print wheel, or character set is available with each printer. The print service treats font cartridges and print wheels the same because they require you to physically mount a new font cartridge or print wheel. Thus, in this discussion, we refer to font cartridges and print wheels simply as font cartridges.

When you list the font cartridges or character sets available, you assign names to them. These names are for your convenience and the convenience of the users on your system. Because different printers might have similar font cartridges or character sets, use common font names on all printers. This allows a user to submit a file for printing and request a particular font style, without requiring that the user know which printer is used or whether a font cartridge or selectable character set is used.

If the printer has mountable font cartridges, you only need to list their names. If the printer has selectable character sets, you must list their names and map each set to a name or number that uniquely identifies the set in the *terminfo* database.

See also:

- "Specifying character sets" (this page)
- "Specifying font cartridges to use with a printer" (page 199)

Specifying character sets

For printers that allow selectable character sets, determine the names of the character sets and then map each set to a name or number in the *terminfo* database.

To determine the names of the character sets listed in the *terminfo* database, enter:

TERM=*printer-type* tput csnm 0

printer-type is the name of the printer type in question. This command should display the name of the 0th character set (the character set obtained by default after the printer is initialized).

To display the names of the other character sets, repeat the command above, replacing **0** with 1, 2, 3, and so on.

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.

To specify a list of character set names and to map them into *terminfo* names or numbers, enter:

/usr/lib/lpadmin -p *printer_name* -S *characterset_list*

characterset_list is a list of names, separated by commas or spaces. Each item in the list is a character set name "mapping" (alias) that looks like one of the following:

cs*N*=*characterset_name*
characterset_name1*=*characterset_name2

N is a number between 0 and 63 that identifies the number of the character set in the *terminfo* database. ***characterset_name1*** identifies the character set by its name in the *terminfo* database. In both instances, 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 *terminfo* name, by number, or by your alias.

For example, 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*. 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", which are the names of the selectable character sets, are the output of the commands.

To change the names of the standard set and set #1, enter:

/usr/lib/lpadmin -p *printer_name* -S "cs0=american, english=british"

A printer that has selectable character sets can accept any **cs*N*** name or *terminfo* name known for the printer.

To remove the character set mappings, enter:

/usr/lib/lpadmin -p *printer_name* -S none

See also:

- "Font cartridges, character sets, print wheels" (page 197)
- **lpadmin**(ADM) manual page

Specifying font cartridges to use with a printer

Until you specify the font cartridges (or print wheels) that can be used with a new printer, the print service does not consider any font cartridges installable on that printer and rejects any print requests that require a font cartridge.

To specify a list of font cartridges to use with a printer, enter:

/usr/lib/lpadmin -p *printer_name* **-S** *font_cartridge_list*

font_cartridge_list is a list of font cartridge names, separated by commas or spaces. If you use spaces to separate the names, enclose the entire list (but not the **-S**) in quotes. These are the only font cartridges considered installable on the printer.

To remove the font cartridge list from the printer, enter:

/usr/lib/lpadmin -p *printer_name* **-S none**

Once you specify the list of font cartridges installable on the printer, you can install them. See "Changing a font cartridge on a printer" (this page).

See also:

- "Font cartridges, character sets, print wheels" (page 197)
- **lpadmin**(ADM) manual page

Changing a font cartridge on a printer

Before the print service prints a file that requires a font cartridge (or print wheel), you must install and mount the font cartridge on the printer.

If you have set up alerting for the font cartridge, the print service alerts you when enough print jobs are queued for the font cartridge to be installed and mounted. See "Alerting to mount forms and font cartridges" (page 203).

Changing a font cartridge involves first removing the current font cartridge from the printer. Then, install the new font cartridge on the printer and inform the print service that the new font cartridge is ready to use by mounting it. Because it is difficult to do this on a printer that is currently printing and because the print service continues to print files that do not require the font cartridge on the printer, disable the printer first.

To install or change a font cartridge:

1. Disable the printer (page 152).
2. Remove the current font cartridge from the printer (if applicable).
3. Install the new font cartridge on the printer.
4. Mount the new font cartridge by entering:

   /usr/lib/lpadmin -p ***printer_name*** **-M -S** ***font_cartridge_name***

   Any print requests that require a font cartridge are printed on ***printer_name***.
5. Reenable the printer (page 152).

To unmount a font cartridge, enter:

/usr/lib/lpadmin -p ***printer_name*** **-M -S none**

NOTE You do not need to unmount the current font cartridge after physically removing it from the printer before installing and mounting a new font cartridge.

See also:

- "Specifying character sets" (page 197)
- **lpadmin**(ADM) manual page

Setting up printer fault alerts

The print service provides a method for detecting and alerting you to printer faults. Faults can range from simple problems, such as running out of paper, ribbon, or 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 the carrier (the signal that indicates that the printer is online) to sending an XOFF or a message.

The print service itself only recognizes two classes of printer fault indicators: "hangups" (a loss of carrier) and excessive delays in printing (an XOFF flow-control character without a matching XON). For faults other than these, the printer service cannot determine the cause of the fault, so it cannot alert you. However, you can add filters that can detect other printer faults and inform the print service, which in turn alerts you. See "Detecting printer fault indicators with filters" (page 196).

To set up printer fault alerts for a printer in the **Printer Manager** (page 142):

1. Select the printer from the list.
2. Select **Advanced** from the **Settings** menu, then select **Faults**.
3. Select the "Fault Recovery" method. See "Specifying the print fault recovery method" (page 202).
4. Select the "Notification Method" (how or if the print service alerts you when an error occurs):

No notification

Specifies that the print service not send any alerts when a fault occurs.

> **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.

Quiet current fault notification

Once a fault occurs and you start receiving repeated fault alerts, directs the print service to stop sending alerts.

Execute command

Specifies that the print service run a command for each fault alert. Enter the command to run in the "Command" field.

The print service runs the command you specify using the shell environment currently in effect when you start the **Printer Manager**. This environment includes environment variables, user and group IDs, and the current directory.

Write to screen

Specifies that the print service write a message to your terminal. See the **write**(C) manual page.

Mail administrator

Specifies that the print service send electronic mail to the print service administrator. See the **mail**(C) manual page.

5. Specify the "Frequency" (the number of minutes between repeated alerts). To receive only one alert for each fault that the print service detects, select **Once**.

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. The print service assumes that a fault is cleared when it successfully prints a file. Until that time, if you have requested only one alert per fault, you do not receive another alert.

If, after you have fixed a fault but before the print service tries to print 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 reenabling of the printer overcomes the problem. Refer to "Specifying the print fault recovery method" (this page) for more information.

See also:

- "Detecting printer fault indicators with filters" (page 196)

Specifying the print fault recovery method

To specify how the print service recovers once the printer is ready for printing again after detecting a printer fault, in the **Printer Manager** (page 142) select **Advanced** from the **Settings** menu, then select **Faults**. Choose the "Fault Recovery" method:

Disable printing	Disables the printer until the administrator reenables the printer.
Restart current job	Restarts printing at the beginning of the print job that was active when the fault occurred.
Continue with current page	Continues printing at the top of the page where printing stopped.

> **NOTE** This method requires a filter that can wait for a printer fault to be cleared before resuming properly. The default filter that the print service uses cannot do this.

If the recovery method is **Continue with current page**, but the interface program does not continue running and therefore cannot detect when the printer fault was cleared, the print service attempts to print every few minutes until it succeeds. You can force the print service to retry immediately by reenabling the printer. See "Enabling and disabling printers" (page 152).

If you do not specify how the print service is to resume after a printer fault, the print service tries to continue at the top of the page where printing stopped (if the appropriate filter exists). Failing that, the print service restarts printing at the beginning of the print job.

See also:

- "Setting up printer fault alerts" (page 200)
- **lpadmin**(ADM) manual page (for command-line interface)

Alerting to mount forms and font cartridges

If you have printers that accept changeable font cartridges and you have listed the font cartridges allowed on each, users can submit a print request to use a particular font cartridge. However, if the font cartridge is not mounted when a user requests to use it, the job waits in the queue until you mount the font cartridge. See "Specifying font cartridges to use with a printer" (page 199). If a form (or font cartridge or print wheel) is not mounted when you print a file and specify that form, the job waits in the queue until you mount the appropriate form. See "Mounting and unmounting forms" (page 191).

In these cases, you might want to set up the print system to alert you when you need to mount a form. You can specify that you are to receive alerts when the number of requests waiting for a font cartridge or form exceeds some threshold.

To arrange for alerting to the need to mount a form, enter:

/usr/lib/lpforms -f ***form_name*** **-A** ***alert_method*** **-Q** ***number*** **-W** ***minutes***

alert_method	alerting method to use (**mail** or **write**)
number	number of waiting requests to restart alerting
minutes	number of minutes between alerts

For example, to direct the print service to send electronic mail alerts every 5 minutes whenever the printer queue contains 2 or more requests for the *check* form and it is not already mounted, enter:

/usr/lib/lpforms -f check -A mail -Q 2 -W 5

To arrange for alerting to the need to mount a font cartridge, enter:

/usr/lib/lpadmin -S ***font_cartridge_name*** **-A** ***alert_method*** **-Q** ***number*** **-W** ***minutes***

For example, to direct the print service to write alerts to your terminal every 2 minutes whenever the printer queue contains three or more requests for the *dingbat* font cartridge and it is not already mounted, enter:

/usr/lib/lpadmin -S dingbat -A write -Q 3 -W 2

To arrange for alerting whenever the queue contains requests for any form or font cartridge (print wheel), enter one of the following:

/usr/lib/lpforms -f any -A mail -W 5

/usr/lib/lpadmin -S any -A mail -W 5

To stop receiving alert messages to mount a form, font cartridge, or print wheel, enter one of the following:

/usr/lib/lpforms -f ***form_name*** **-A quiet**

/usr/lib/lpadmin -S ***font_cartridge_name*** **-A quiet**

/usr/lib/lpadmin -S ***printwheel_name*** **-A quiet**

To remove alerting when a form, font cartridge, or print wheel needs to be mounted, enter one of the following:

/usr/lib/lpforms -f ***form_name*** **-A none**

/usr/lib/lpadmin -S ***font_cartridge_name*** **-A none**

/usr/lib/lpadmin -S ***printwheel_name*** **-A none**

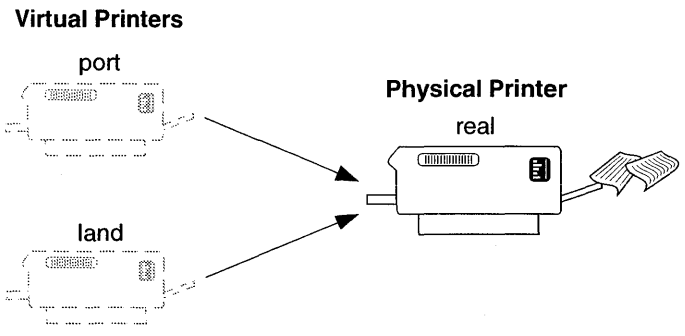
If you do not define an alert method for a form, font cartridge, or print wheel, you do not receive an alert for it. If you define a method, but do not define the number of minutes between alerts (with the **-W** option), you receive one alert for each occasion.

See also:

- **lpforms**(ADM) manual page
- **lpadmin**(ADM) manual page

Setting up a printer with multiple names

The print service allows you to set up a single printer with multiple names to perform multiple functions. For example, if your printer supports both portrait and landscape modes, you can set up a different name for each function and then send jobs to each printer name. These multiple printers are called "virtual printers".



The print spooler system differentiates printers by name only, not by the device to which the printer is connected. To specify different names for the same device and prevent multiple print jobs from appearing simultaneously, you must set up both the *real* printer and additional *virtual* printers. The real printer performs the actual printing and the virtual printers pass print jobs to the real printer.

For example, to set up two virtual printers, *port* and *land*, that use the capabilities of an Hewlett-Packard LaserJet, use this procedure:

1. Set up the real printer. See "Adding local printers" (page 144). Specify the "Name" as *real* and set the "Model" to *HPLaserJet*.
2. Set up the *port* and *land* virtual printers. Set the "Model" to *network*. Specify the same "Device" to which *real* is connected.
3. Create the file */usr/spool/lp/remote* and add the following lines:

```
port:   lp -dreal -oportrait
land:   lp -dreal -olandscape
```

   This specifies that when printing to printer *land*, the print system sends the print job to printer *real* using the **-olandscape** option (to print in landscape mode) and when printing to printer *port*, the print system sends the print job to printer *real* using the **-oportrait** option (to print in portrait mode)

> **NOTE** The options listed after **-dreal** depend on the printer model. Check the interface script in */usr/spool/lp/admins/lp/interface* for your printer to determine the printer- or class-dependent **-o** options.

To print a file in landscape mode, enter:

lp -dland *filename*

This procedure translates the **-dland** option to **lp** to the necessary options for the printer (in this example, **-dreal -ol**).

Another way to accomplish this is to create a simple shell script to perform the type of printing. For example:

```
:
# Land - shell script to print in landscape mode
#
# syntax:  land <file> <file> ...
#
#
lp -dreal -ol $@
```

The reasons for choosing one method over the other depend on how your applications access the print system. Many applications allow you to specify only the name of the printer, so virtual printers are the only solution. Other

applications might allow complete control over the commands that submit the print job. In this case, you might use the shell script above.

See also:

- **lp**(C) manual page (for command-line interface)
- **lpadmin**(ADM) manual page (for command-line interface)
- "Adding local printers" (page 144)

Attaching a printer to a serial terminal

The operating system supports the use of local printers attached to the AUX or PRINT port on the back of many serial terminals. You can connect these printers via standard RS-232 connections to reduce the load on shared system printers. Once you configure the terminal properly, you can use the **lprint**(C) command to print files on the local printer.

NOTE Your terminal's **stty** settings must match those of the printer. (In most cases, this is true.) If the terminal and printer require different **stty** settings, see "Handling different stty settings" (page 209).

To add a printer connected to the AUX or PRINT terminal port:

1. Connect your local serial printer to the AUX port on your terminal with a standard RS-232 cable with at least pins 2, 3, and 7 connected. Make sure the printer is powered on and online. (If the terminal supports pass-through mode to the parallel port, you can use the parallel port.)
2. Log into the system on the terminal and verify that the terminal is working correctly.
3. Make sure that the AUX port on your terminal is configured with the same settings as your printer (baud rate, parity, data bits, XON/XOFF, and so forth).
4. Verify that the entry for your terminal in */etc/termcap* (or an alternate file, defined by the **TERMCAP** variable) includes definitions for PN (start printing) and PS (stop printing).

   The **lprint** command must know how to start and stop local printing for each specific terminal. The PN and PS terminal attributes in */etc/termcap* define escape sequences that are sent to the terminal to control local printing.

   NOTE Very few terminals have these attributes defined in their *termcap* entries.

   Use a text editor to search for the entry for your terminal in the */etc/termcap* file. For example, if your terminal is a Wyse 60, search for "wyse60".

Example 4-1 Wyse 60 termcap entry

```
w7|wy60|wyse60|Wyse WY-60 with 80 column/24 line screen in wy60 mode:\
    :is=\E'\072\Ee(\E0\Ee6\Ec41\E~4\Ec21\Ed/:\
    :if=/usr/lib/tabset/std:pt:\
    :G1=\EH3:G2=\EH2:G3=\EH1:G4=\EH5:GD=\EH0:GG#0:GH=\EH\072:\
    :GU=\EH=:GV=\EH6:GR=\EH4:GL=\EH9:GC=\EH8:GF=\EH7:\
    :PU=\EJ:PD=\EK:\
    :al=\EE:am:bs:bt=\EI:cd=\EY:ce=\ET:cl=\E+:\
    :cm=\Ea%i%dR%dC:co#80:dc=\EW:dl=\ER:ei=\Er:im=\Eq:k0=^AI\r:\
    :k1=^A@\r:k2=^AA\r:k3=^AB\r:k4=^AC\r:k5=^AD\r:k6=^AE\r:k7=^AF\r:\
    :k8=^AG\r:k9=^AH\r:kd=^J:kh=^~:kl=^H:kr=^L:ku=^K:\
    :li#24:mi:nd=^L:se=\EG0:so=\EG4:sg#0:ug#0:ue=\EG0:ul:up=^K:us=\EG8:\
    :PN=\Ed#:PS=^T:hs:ts=\Ez(:fs=^M:
```

The Wyse 60 has PN and PS defined.

5. If *termcap* does not define PN and PS for your terminal, you must add a line with the following format:

   :PN=*start sequence*:PS=*stop sequence:*

   Log in as *root* to edit */etc/termcap*. Make a backup copy of */etc/termcap* before editing it.

   Refer to your terminal manual to find the sequence of control characters used to switch the auxiliary port on and off (this is sometimes referred to as "passthrough" or "transparent" mode). For example, for a Wyse 60 terminal, the code to switch on the port auxiliary printing is:

   ⟨Esc⟩ **d #**

   And the code to turn it off is:

   ⟨Ctrl⟩**t**

   You must translate these keystrokes into *termcap* format before inserting them into the *termcap* file. *termcap* uses the codes in Table 4-9 (this page) to represent keystrokes.

Table 4-9 termcap keystroke translations

Keystroke	termcap sequence
ESCAPE	\E
CTRL x	^x (x is any character)
NEWLINE	\n
RETURN	\r
TAB	\t
BACKSPACE	\b
FORMFEED	\f

NOTE To use a control sequence, use the caret (^) symbol, not the ⟨Ctrl⟩ key. For example, ⟨Ctrl⟩**x** translates to ^x.

You can represent characters by their caret (^) and backslash (\) characters or by their octal codes. See the **ascii**(M) manual page. Separate *termcap* attributes by a colon (:).

Translate the *termcap* entry for the Wyse 60 keystrokes for PN (⟨Esc⟩ **d #**) and PS (⟨Ctrl⟩**t**) into a *termcap* entry.

```
:PN=\Ed#:PS=^T:\
```

For a terminal missing these entries, simply insert a modified version of this line into the *termcap* entry for the terminal. For other terminals, check your owner's manual and locate the proper sequences for turning the auxiliary print mode on and off and substitute the *termcap* sequences as in the example. Some terminals (such as the Wyse 60) include a transparent mode, where the data is not displayed on the screen as it is printed. This is the mode selected by the PN sequence in the example.

6. Once you add the PN and PS entries, log out of and back into the terminal to activate the new *termcap* entry.
7. Use the following command to print the file ***filename*** on your new local printer:

   lprint ***filename***

   Do not touch your keyboard while local printing is taking place. You cannot perform other tasks on your terminal while printing.

If your file prints on the screen instead of on the printer, the PS and PN entries you created are incorrect. Revise the entries with the correct codes. If the file still does not print on the printer or the terminal, try crossing the Transmit and Receive Data pins in the cable connecting the terminal AUX port and the printer. (This is also known as a "null modem" connection.)

To eliminate carriage return delays, set the environment variable **CRDELAY** to "N". Do this only if you are running a fast printer.

If, when printing with **lprint**, everything prints on one line, set the environment variable **FORMS** to "X" to turn on **lprint**'s "transparent" mode. In this mode, **lprint** does not perform special processing of carriage returns, line feeds, form feeds, or tabs.

- In the Bourne or Korn shells (**sh** or **ksh**), use:

  FORMS=X; export FORMS

- In the C shell (**csh**), use:

  setenv FORMS X

Put these definitions either in individual users' *.profile* or *.cshrc* files or in the */etc/profile* or */etc/cshrc* files. If you add them to */etc/profile* and */etc/cshrc*, they affect all users on the system.

Handling different stty settings

If the terminal and printer require different **stty** settings, you must create a shell script to handle that:

1. Create a file called *lprints* and enter the following text:

```
:
# /usr/bin/lprints
#
# Do local printing with stty settings that differ from
# those of the terminal. The required settings are read
# from environment variable LPRINTSTTY.

oldstty=`stty -g`
[ "$LPRINTSTTY" != "" ] && stty $LPRINTSTTY

FORMS=X /usr/bin/lprint "$@"

stty $oldstty
```

2. Change the permissions on the file:

   chmod 755 lprints

3. Put the file in an appropriate directory (for example: */usr/bin*) and use it instead of **lprint**:

   lprints *filename*

   Users can store the **stty** settings required for a local printer using the environment variable **LPRINTSTTY**. You can set system-wide values in the */etc/profile* and */etc/cshrc* files:

```
LPRINTSTTY="ixon ixoff -ixany onlcr"; export LPRINTSTTY
```

See also:

- **termcap**(F) manual page

Configuring a spooled local terminal printer

To configure a spooled local terminal printer using an intelligent serial card with built-in support for local printing, configure the printer using one of the special device nodes provided with your card.

If you are using a dumb serial port or if the device driver for your intelligent serial port does not provide special handling for local terminal printers, then you must use a special null device name. When you stop I/O on this device, no other processes are affected.

To set up the spooled local terminal printer:

1. Log in as *root*.
2. Create a second null device. Enter these commands:

   cd /dev
   mknod lpnull c 4 128
   chown lp lpnull
   chgrp lp lpnull
   chmod 660 lpnull

   Use same major number as */dev/null* for *lpnull* (4 in this case) and use 128 for the minor number. For more information, see the **mknod**(C) manual page.
3. Create a copy of the printer interface script that you want to use:

   cd /usr/spool/lp/model
   copy -om *interface_script new_script*
   chmod u+w *new_script*

   interface_script is the name of the printer interface script to use; ***new_script*** is the copy.
4. Edit ***new_script*** to:
   a. Change the name in the comment at the top of the script to indicate the name of the new interface.

   NOTE The following sequences are for a Wyse 60 terminal. If you are using a different type of terminal, refer to your terminal manual for the escape sequences to lock and unlock the terminal's keyboard and to turn on and off transparent print mode, and use them instead.

b. Add the following code before any I/O occurs:

```
# setup string termport="/dev/tty2h"
# change tty2h to port for your terminal penable="\017\033d#pdisable="
  "\024 161pnull="/dev/lpnull" codes to enable printer for Wyse 60 pdisable=
  "\024\016\c"
# codes to disable printer for Wyse 60 lpnull="/dev/lpnull"
# dummy regular file or dedicated device file sttystr="ixon -ixany
  ixoff opost onlcr"
# modify for your printer if necessary

# redirect std in, std out to termport  std err to lpnull #sleep 3
# uncomment this is if terminal data appears on your printout
  exec <$termport >$termport 2>$lpnull

# If it is necessary to change the baud rate or other stty settings for
  your serial printer modify the sttystr above.  sttysave='stty -g'
  stty $sttystr

# escape sequence for the terminal to lock the keyboard and turn
  on transparent print mode.
# You must first lock the keyboard then turn on transparent print
  echo "$penable"
```

c. Add the following code at the end of the script immediately before the exit:

```
# escape sequence for the terminal to turn off transparent print
  mode and unlock the terminal.  First turn off transparent print
  then unlock keyboard.
# sleep 2
# uncomment this if print data appears on your terminal echo
  "$pdisable"

# reset stty settings stty $sttysave
```

5. Use the **Printer Manager** (page 144) to set up your local terminal printer.
6. If you are setting up multiple spooled local terminal printers, set up each printer as described in step 5 (this page). Otherwise, skip to step 7 (page 212).

   After setting up multiple spooled printers, enter these commands:

   cd /dev
   rm lpnull
   touch lpnull
   chown lp lpnull
   chgrp lp lpnull
   chmod 660 lpnull

These commands replace the character device special file */dev/lpnull* with an empty regular file of the same name. This regular file can be shared by the spooled local terminal printers without causing the problems that would occur if they shared a device special file.

7. If it is not already running, start up the print service (page 153).
8. Enable the new printer (page 152).
9. Set up the printer to accept jobs (page 152).
10. Test the new local terminal printer. Log into the terminal and enter:

    lp -d*printer_name* /etc/motd

    The message of the day should print on the local terminal printer.

Only two user IDs can use the local terminal printer: the user ID logged into the **termport** device and *root*. To allow others to access the printer temporarily:

disable tty*xx*
chgrp lp /dev/tty*xx*
chmod 660 /dev/tty*xx*

Replace *xx* with the **termport** device name.

To return the port to normal use, enter:

enable tty*xx*

Replace *xx* with the **termport** device name.

Initializing parallel printers with an init device file

The standard parallel printer devices (*/dev/lp*, */dev/lp0*, */dev/lp1*, and */dev/lp2*) send a printer initialization string (**init**) the first time, and only the first time, the device is opened after the system starts up.

Some parallel printers require initialization every time a file is received for printing. Others require an **init** if the printer is turned off and back on again (for example, after changing paper or ribbon).

If you need to initialize the printer more often than the standard devices provide, you can create an additional device file for the parallel port. Then, use this **init** device file to initialize the printer when necessary.

1. Log in as *root*.
2. Determine which device is the parallel port you are using. In this example, the device is the main parallel port (*/dev/lp0*).
3. Enter:

    /usr/lib/lpadmin -p *printer_name* -v /dev/lp0i

    This associates the parallel **init** device, */dev/lp0i*, with ***printer_name***.

4. If your printer requires an **init** when you turn it off and on after turning on the printer, initialize the printer before the first file is sent to the printer (this example assumes the main parallel port) by entering:

   > /dev/lp0i

   If your printer requires an **init** every time a file is sent (and it does not have a large internal text buffer), you can use the */dev/lp0i* device all the time. The **lp**(C) command then sends an **init** every time you send a file to the printer.

See also:

- **lpadmin**(ADM) manual page

Customizing the toolbar

In the **Printer Manager** (page 142) and **Print Job Manager** (page 168), you can customize the toolbar by rearranging and deleting icons, and inserting spaces to change the grouping.

To customize the toolbar, select **Customize Toolbar** from the **Options** menu. To effect your changes, click on **OK**.

To delete an icon or space, select the icon or space from the "Toolbar" list, then click on the **Delete** button.

To add an icon:

1. Select a position in the "Toolbar" list (from left-top to right-bottom).
2. Click on the appropriate **Add** button. For example, to add an icon to the left of the position in the "Toolbar" list, click on **Add Button Left** button.
3. Select an icon from the "Icon" list.
4. Select a command from the "Command" list to associate with the icon you selected. The "Description" field describes the selected command.

NOTE You cannot add a command that already exists in the toolbar.

To add a space, select a position in the "Toolbar" list and click on the appropriate **Add** button. For example, to add a space to the left of the position in the "Toolbar" list, click on **Add Space Left** button.

You can restore the original toolbar with the **Restore Default Toolbar** button.

The **Change Border** button toggles the style of the toolbar border between a line and a shaded line.

Troubleshooting the Printer Manager

Most error messages displayed by the **Printer Manager** are self-explanatory. In other cases, the error message will be generic, as in these examples:

```
Failed to establish user authorizations

Failed to retrieve list of local printers and their descriptions
```

Error boxes of this type include a **Details** button to provide you with additional information. The problems reported fall into these categories:

- Remote administration problem (this page)
- Missing or corrupted database files (this page)

Remote administration problem

If you are performing remote administration, the remote system may be unreachable or there may be a configuration problem (including the lack of user equivalence):

localhost `failed to connect to remote host... Error with server process: Permission denied.`
These messages (and suggested solutions) are displayed when an **Open Host** operation fails. Check to make certain you have user equivalence on the remote host. See "Adding user equivalence" in the *Networking Guide* for more information.

`Failure with connection to server`

`Error with server process:` ***remotehost***`: Connection timed out`
The remote host could not be reached. Check to see if the system is up and on the network.

For more information on network problems, see "Troubleshooting TCP/IP" in the *Networking Guide.*

Missing or corrupted database files

If certain files used by the **Printer Manager** have been removed or corrupted, the manager may fail to function:

/etc/printcap
If this file is missing, the **Printer Manager** will become stuck during initialization and will not start up. The progress indicator will work continuously until the screen is closed. If the file is corrupted, an error message will be displayed. You should restore the file from backups.

*/usr/spool/lp/admins/lp/printers/**printername**/configuration*
If the configuration file is missing for a given printer, the **Printer Manager** will abort at startup with an error trace saved to */tmp/tclerror.**number**.log*. Check the files in the above directory to see if any are missing. If there are, restore them from backups.

Troubleshooting the print system

This section covers problems you might encounter while using your printer:

lpsched print scheduler is not running

If the print service stops in the middle of a print job or does not start any new print jobs, determine if the **/usr/lib/lpsched** daemon is running. To do this, in the **Printer Manager** (page 142) select **Print Services** from the **System** menu. The **lpsched** daemon is started automatically by a script in the */etc/rc2.d* directory at system startup.

If the scheduler is down, **Local print service enabled** is deselected. To start the **lpsched** daemon, select **Local print service enabled**.

NOTE You must be *root* to start the print service.

See also:

- "Starting and stopping the print services" (page 153)

Printer does not print

If the printer is sitting idle and there is no output, check the following:

1. Make sure that the printer has power.
2. Check that the printer is not out of paper, ink, or toner and that it is not in an error state.
3. Verify that the printer hardware is working before continuing. Run a self-test (refer to the printer manual for instructions).
4. Check the printer cable and make sure that it is attached properly to the port and the printer. Refer to the printer manual for installation instructions.
5. Make sure that the printer is configured properly. To set up your parallel or serial printer to receive data properly, follow the instructions in Chapter 22, "Adding printers" in the *SCO OpenServer Handbook.*

   If the printer is a serial printer, make sure that the baud rate at which the computer sends data to the printer matches the printer's baud rate. For instructions on how to reset the baud rate, see "Printer output is illegible" (page 218).
6. Make sure that the printer is enabled and accepting jobs. See "Enabling and disabling printers" (page 152) and "Accepting or rejecting print jobs" (page 152).
7. Verify that the system has the port configured properly. Use the **hwconfig**(C) command or check the */usr/adm/messages* file for these messages (for parallel and serial, respectively):

   ```
   %parallel 0x378-0x37A   07    -    unit=0
   %serial   0x03F8-0x03FF 04    -    unit=0 type=Standard nports=1
   ```

   If you do not see a similar message for the printer port, refer to "Port does not respond" (page 218) for more information.
8. Make sure that the port is configured for the proper interrupt vector and that no other hardware is using that interrupt vector. See "Typical device interrupts" in the *SCO OpenServer Handbook* for information on the available interrupt vectors. See your hardware documentation for information on configuring your ports.
9. Test the printer port connection by redirecting the output of a command directly to the device.
   - For parallel printer *lp0*, enter:

     date > /dev/lp0

   - For serial printer *tty1a*, enter:

(stty *options*;date) > /dev/tty1a < /dev/tty1a

where ***options*** are baud rate, parity, or other settings that you want to pass to the serial printer. These options should include **opost** and **onlcr**. See the **stty**(C) manual page.

If the output from the redirected command does not print, follow the instructions in "Cannot redirect output to printer" (this page).

If the output prints, try submitting a sample file for printing:

lp -d*printer_name filename*

If the hardware connections are good and the printer is properly configured and enabled, but is still idle and print jobs are queued:

1. Verify that **lpsched** is running. See "lpsched print scheduler is not running" (page 215).
2. Restart the **lpsched** daemon if necessary. See "Starting and stopping the print services" (page 153).
3. Check to see that print jobs are being queued:

   lpstat -o -l

   This command displays a detailed description of the status of output requests, printer names, and devices.
4. If the printer detects a fault, it does not immediately continue automatic printing. Force a retry by enabling the printer. See "Enabling and disabling printers" (page 152).
5. If applicable, check to see if a dialout printer was busy or did not answer, or all dialout ports are busy. The print service waits 5 minutes before trying to reach a dialout printer again. Force a retry by enabling the printer.

Cannot redirect output to printer

If you redirect output directly to the parallel or serial port and nothing happens or the system displays the `cannot create` message:

1. Verify that the device file for the port exists in */dev*. Make sure that this file is a device file and not a text file. For example, use the following command to check */p0*:

   ls -l /dev/lp0

   The output should look similar to this:

   ```
   crw-------   2 bin      bin        6,  0 Jun 13 1993 /dev/lp0
   ```

   A regular text file does not include the " c " character (for character special file) at the beginning of the line.

2. Test the cable connection using a cable from a working device with the same cabling requirements.
3. Print a file while the system is running under DOS. If you can print a file under DOS but not under the UNIX system:
   a. Verify that the port is configured correctly. See "Port does not respond" (this page).
   b. If the port configuration is correct and you still cannot redirect output to the port, try using a different device name. For example, for a parallel port, use *lp1* instead of *lp0*. For a serial port, use *tty2a* instead of *tty1a*.

      Pay attention to the `unit=` message for parallel ports on the boot screen or use **hwconfig** (for example, `unit=0` corresponds to *lp0*).
   c. If you still cannot print using a different device name, your printer may be defective; check the printer documentation.
4. If you cannot print from DOS, check the printer hardware configuration. See the documentation provided with your printer.

If you configured your printer correctly and you still cannot redirect output to it, the problem is most likely a hardware malfunction. Recheck the cables and port configuration and consult your hardware documentation.

Port does not respond

If your serial or parallel port does not respond:

1. Verify that the parallel or serial card is properly seated:
   a. Turn the power off and open the machine.
   b. Remove and reseat the card in the bus.
2. Run the **Hardware/Kernel Manager** to configure the serial port and parallel port drivers and verify that the port is listed in your kernel configuration. See "Configuring drivers with the Hardware/Kernel Manager" in the *SCO OpenServer Handbook*.
3. Verify that the card itself is correctly configured. Check the documentation provided with the card. If possible, try setting the card for a different configuration.
4. The card may be defective. Try replacing it.

Printer output is illegible

If the printer prints illegible output:

1. Determine the baud rate for the serial printer and check to see that it matches the baud rate for the computer. (No baud rate is associated with a parallel port, although **stty**(C) does display one.)

   To change the baud rate for a serial printer, in the **Printer Manager** (page 142) select **Serial Comm** from the **Settings** menu, then select a new baud rate.

   Print a sample file.

2. Verify that the parity setting matches the computer's parity setting. (If the printer is directly connected to the computer with a wire that is less than 50 feet long, it does not have to use the parity bit.)

   To set the parity bit, in the **Printer Manager** (page 142) select **Serial Comm** from the **Settings** menu, then select a new parity setting.

3. Check to see that the tabs are set correctly. See "Printer output spacing is wrong" (this page) for more information.

If the settings and baud rate are correct and the output is still illegible, check to see that the printer model is correct. If you set up the printer with the wrong model name, the wrong control characters can be sent to the printer. This situation can cause output to disappear or be illegible.

To verify and change the printer model, in the **Printer Manager** (page 142) select **Model** from the **Settings** menu, then select a new printer model.

Printer output spacing is wrong

If the printer output is legible, but the spacing is wrong, adjust the stty settings:

/usr/lib/lpadmin -p *printer_name* **-o stty=***'stty_setting'*

For each of the situations, adjust the **stty** setting:

- If the printer output is double-spaced, replace *'stty_setting'* with either **'-onlcr'** or **'-tabs'**.
- If there is no left margin and the text runs together, replace *'stty_setting'* with **'-tabs'**.
- If the printer output zig-zags down the page, replace *'stty_setting'* with **'-onlcr'**. (This is set by default, but you may have cleared it accidentally.)

Parallel printer is slow

If your parallel printer prints abnormally slowly, verify that the configuration settings are correct. In particular, printing can be slow if the port is configured for the right I/O address, but the wrong interrupt vector. To check and change this setting, see "Adding and configuring parallel ports" in the *SCO*

OpenServer Handbook. To change the interrupt vector for a slow port, remove and reinstall the parallel port with the same I/O address but a different interrupt vector (5 or 7). Relink the kernel in the *SCO OpenServer Handbook*, reboot the system in the *SCO OpenServer Handbook*, and see if the port is still slow.

If the parallel ports are configured correctly and printing is still slow, your parallel port may not be capable of generating interrupts. In this case, you can try setting up polling (this page) for your printer port.

If your line printer is taking about four seconds per line, the printer may be deselecting itself after receiving each line of text and then reselecting. In this case, you can modify a kernel parameter (page 221) to change this behavior.

Setting up polling

To speed up printing on your parallel printer, you can alter the way that the hardware and the printer driver communicate. The parallel printer driver can be made to "poll" a parallel port so that the driver does not rely on interrupts from the parallel port.

NOTE When the printer driver polls a parallel port, you may experience a drain on system resources.

To set up polling for a parallel port or parallel printer, you must create a polling device file and reconfigure your printer to use it instead of the standard device. Use this procedure:

1. Bring the system into maintenance mode. See "Using the shutdown command line" in the *SCO OpenServer Handbook*.
2. Note which parallel printer ports are configured. Enter:

   /etc/hwconfig -h

   You should see output like this:

   ```
   parallel 0x378-0x37A   07    -    unit=0
   ```

   where `unit=0` refers to *lp0*.
3. Configure a special device file for the printer. Create a file called *pa* (if it does not already exist) in the */etc/conf/node.d* directory containing one of the following lines:

   For *lp0*: `pa lp0p c 64 bin bin 600`
   For *lp1*: `pa lp1p c 65 bin bin 600`
   For *lp2*: `pa lp2p c 66 bin bin 600`
4. Enter these commands to reconfigure the kernel to recognize the new device:

```
cd /etc/conf/cf.d
touch /etc/.new_unix
../bin/idmkenv
```

5. At the prompt to rebuild the kernel environment, enter **y**.
6. Reboot your system. See "Using the haltsys command" in the *SCO OpenServer Handbook*.

   When you enter multiuser mode, the new polling device is in place.
7. If you are using the print spooler, you must now inform the spooler of the new parallel poll device.

   In the **Printer Manager** (page 142), select **Connection** from the **Settings** menu. Do not select a standard parallel device name. Instead, select or enter either */dev/lp0p*, */dev/lp1p*, or */dev/lp2p*.

Changing the MODE_SELECT kernel parameter

If you suspect that your printer is deselecting and reselecting itself after each line, change the value of the **MODE_SELECT** kernel parameter. (This works with both polled and standard **lp** devices.) The default value of **MODE_SELECT** is 1, which enables mode-select checking. To disable mode-select checking, set **MODE_SELECT** to 0. This removes the printing delay incurred by these checks.

To change the value of **MODE_SELECT**, use the **Hardware/Kernel Manager** as described in "Configuring drivers with the Hardware/Kernel Manager" in the *SCO OpenServer Handbook*.

1. Select **Tune Parameters** from the **Kernel** menu.
2. Select **18** (Miscellaneous device drivers and hardware parameters).
3. Press ⟨Enter⟩ until you see **MODE_SELECT**, then enter **0**.

See "Configuration tools" in the *Performance Guide* for complete instructions.

Printer reports UUCP errors

If UUCP is configured, the print service uses the UUCP software to handle dialup printers. If a dialing failure occurs and you receive printer fault alerts, the print service reports the same error reported by the UUCP software for similar problems. (If you have not arranged to receive fault alerts, this information is mailed to the user *lp* by default.) Refer to "Common UUCP error messages" (page 344) and "Troubleshooting UUCP" (page 336) for more information.

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. SCO systems are designed to meet the requirements of the C2 class of trust as defined by the Department of Defense *Trusted Computer System Evaluation Criteria* (also known as TCSEC or the *Orange Book*).

This chapter explains how to use the security features to maintain a trusted system. Features affecting the ordinary user are described in Chapter 9, "Using a secure system" in the *Operating System User's Guide*.

Security maintenance includes:

Understanding system security

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". A trusted system is one that achieves a greater level of control over access to information, providing mechanisms to prevent (or at least detect) unauthorized access, along with additional means to confirm that these mechanisms are functioning properly. The C2 level of trust means that the system is designed to meet specific criteria in its security policy: accountability, assurance, testing, and documentation.

The security features of SCO systems are an extension of features present on most UNIX systems. Full compatibility with existing UNIX system 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 chapter.

At installation time, you were asked to select the security defaults to be used on your system. In addition, you can customize any of the defaults to the needs at your site.

As administrator, your actions are crucial to maintaining a trusted system. You should understand the system's security policy, how it is controlled by system information databases, and how changes you make affect user and administrator actions.

Physical security

The security features of the operating system are useless if your hardware and media are not protected. You must physically protect the computer itself, the distribution media, and any backup media from unauthorized access:

- Keep your system under lock and key when an operator is not present.
- Organize and lock up all backup media.
- Protect communication lines (UUCP, Ethernet, and terminals) from unauthorized access.

Due to the design of the boot staging of most supported hardware platforms, any person with physical access to a system can easily circumvent any and all software security mechanisms. It is best to keep the system in a constantly supervised area. If supervision is not feasible, secure the area in which the computer is located by physically locking the system. Many systems provide key locks that electrically disconnect the keyboard; others provide CMOS password protection. The use of all these features may provide an adequate level of security.

If these measures are not possible, perform regular security checks as described in "Database consistency checking: authck(ADM) and addxusers(ADM)" (page 249) and "Unsupervised physical access to the computer" (page 246). Strict record-keeping and regular backups become more critical in this type of environment.

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 further details and describe maintenance procedures.

Trusted computing base

A collection of software called the "trusted computing base" (TCB) maintains the parts of the system that are related to security. The TCB consists of the UNIX system kernel (the heart of the operating system) and the trusted utilities that reference and maintain relevant security data. The TCB implements the security policy of the system. The security policy is a set of rules that oversee and guard interactions between "subjects" (such as processes, which are programs running on the system) and "objects" (such as files, devices, and interprocess communication objects). At the C2 level, this consists of "discretionary access control" (DAC), discussed later in this section, and object reuse (page 226) (which dictates that information in a storage object must be cleared before allocation). Much of the software that you interact with is part of the system's TCB. SCOadmin provides a menu-driven interface to help you maintain the TCB.

Accountability

On a trusted system, all actions can be traced to a responsible person. Most UNIX systems lack good accountability because some actions cannot be traced to a person. 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 C2 level of trust requires that the TCB be able to identify each user uniquely and thus enforce individual accountability. The concept of user identity is expanded on trusted UNIX systems 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 (DAC) determines whether a user has access to desired data. That information is within an "object" (file, device, and so on) that the user's process is trying to use. On most UNIX systems, object protection is enforced through the relationship between the user and the 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, who can change the protection bits on a file and even give the file away (change ownership). Your SCO system extends the standard discretionary access control rules used by the UNIX file permissions by restricting the:

- ability to set the SUID and SGID (set user or group ID on execution) bit on files.
- ability to change ownership of files with **chown**(C).
- potential misuse of SUID, SGID, and *sticky** permissions by clearing these bits whenever a file is written.

Object reuse

As required by the C2 level of trust, SCO systems always clears the information in a storage object (memory, whether in RAM or secondary storage) before re-allocating the resource. This prevents users from gaining access to residual data belonging to other users.

Authorizations and privileges

Authorizations and privileges are user attributes required to perform certain actions. Most UNIX systems make all access decisions based on the simple file permissions or on whether the process making the access is owned by *root*. The *root* account can perform system actions that no other process can. System privileges are associated with processes and 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.

* This term comes from an earlier versions of UNIX systems where this permission bit (also called text-save) caused a file to be held in memory. Thus the expression "sticky."

In this way, authorizations allow you to assign "administrative roles": users entrusted with administering specific subsystems (such as the print service).

Privileges are stored in an "privilege set" associated with every process. Thi is a list of privileges which allow a type of action if the privilege is present, and do not allow the action if the privilege is absent. Privileges are either set by the system defaults or defined for a specific user.

Identification and authentication (I&A)

When a user logs into a non-trusted UNIX system, limited identification and authentication (I&A) 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 may be enforced, but these rules have been shown to be insufficient to guard against penetration.

SCO systems extend the standard UNIX system 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 differs from that of other UNIX systems. The administrator also has greater control over the login process. A separate role, called authentication (or accounts) administrator (with subsystem authorization **auth**), is charged with account administration (page 7).

Auditing

Most UNIX systems keep a limited record of system actions with their accounting subsystem. The accounting subsystem writes a single accounting record upon completion of each user process. SCO systems provide an extensive series of records that form a "trail" of actions. In this trail is a record of every access between subject and object (successful and unsuccessful) and every change of subject, object, and system characteristics. The audit subsystem is controlled by a separate role called audit administrator (with subsystem authorization **audit**). The audit administrator decides how much information is recorded, and 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 it occurred, and who was involved.

Protected subsystems

UNIX systems provide the SUID and SGID mechanisms (page 235) that allow programs to set the user of group ID of a process with the **setuid**(S) and **setgid**(S) system calls and permission bits described in **chown**(C). With these you can construct programs that maintain protected information. This information can only be accessed or modified by the operations implemented in the programs. The TCB 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 trusted system extends the notion of a protected subsystem in several ways:

- It provides more precise control of users and groups who 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. (If users log in to anonymous accounts to perform system administration, there is no way to determine who performed a given action.)

Security in a networked environment

The UNIX operating system was originally designed for stand-alone machines, and includes effective security mechanisms for that environment. It also provides flexible mechanisms for giving privileges to application programs. Misuse of this flexibility by privileged programs can result in reduced system security.

Additional security issues arise when a machine running a UNIX system is connected to a network. Much of the current network software design assumes what is called a "single administrative domain." This means that the design assumes that all machines with access to the network are maintained and controlled by a single group of trusted people.

In enterprise networks, where various groups of computer systems are administered and controlled by differing groups of people, these assumptions are no longer valid. This makes the security of the system more difficult to maintain. The computer industry is working to address these issues and expects to see rapid improvement in the security of enterprise networks.

The material in this section is intended to help system administrators to understand the security issues arising from connecting SCO systems to enterprise networks and to provide advice on how to maintain security. The intent is to explain which system facilities might be vulnerable to attack by a "hostile user", and the actions you can take to reduce the risk.

More information on identifying potential security problems is available in these books:

UNIX System Security
Patrick H. Wood and Stephen G. Kochan
Hayden Books, 0-8104-6267-2

Building A Secure Computer System
Morrie Gasser
Van Nostrand Reinhold, 0-442-23022-2

Network Information Service

The Network Information Service (NIS, formerly known as "Yellow Pages") is designed for use on networks consisting of a single administrative domain. However, the protocol used by NIS should not be considered secure even in this type of environment. Due to the design of the NIS protocols, the use of NIS for security information (such as account administration with */etc/passwd* and */etc/group*), is not recommended for any network that connects to systems in another administrative domain. Even then, to prevent abuse of the NIS system, the **ypserv**(NADM) daemon should always be invoked with the **-ypsetme** option. Alternative configurations should be carefully considered by NIS experts prior to installation and configuration.

SCO is working along with the rest of the industry to provide a more secure distributed system management framework than NIS currently provides. In the interim, to minimize the system security risks associated with the use of NIS, we are providing as much functionality as possible while maintaining interoperability with other systems,

The graphical environment

Due to the design of the X11 windowing system, any program that is allowed to access the X display may intercept any and all keyboard events. This, combined with the common use of programs (such as **rlogin**, **su**, or **telnet**) which query the user for password information, can result in giving away passwords to other users.

Users of the X Window System should be very careful about which systems they grant access to their X server. When granting access is combined with the use of X terminals on unsecured networks, serious security problems can arise. Many X terminals default to allowing any and all X clients to communicate with the X display, thus making any system connected to the network a potential source of a security violation.

In previous releases, **xhost** was the only mechanism for controlling access to the server. **xhost** allows you to specify what machines are allowed to connect to your server, which is acceptable if you trust everyone who has an account on the client system.

SCO systems support the MIT-MAGIC-COOKIE-1 Authorization Protocol, which is implemented by the X server, **scologin**, **Xlib**, and the **xauth**(X) client. Under this protocol, when a user uses **scologin** to log in to the X server, **scologin** adds a random access code called a "magic cookie" to a file in the user's home directory called *.Xauthority*. It also passes the magic cookie to the server where it is stored in an area that can only be read by the server. The user's *.Xauthority* file now contains the information needed to connect to the server. Normally, only the user has read and write permissions on this file.

When a client attempts to connect to the server, it must first read the user's *.Xauthority* file and pass the magic cookie to the server in order to obtain a connection. If the magic cookie is incorrect, the server refuses that connection. If the correct magic cookie is presented, then the server accepts the connection, regardless of where the client is actually running.

The **xauth** program allows you to edit the *.Xauthority* file and can be used in conjunction with **rcp**(TC) to pass the magic cookies to other systems and users.

To set up the Xauthorization mechanism for the system, edit the file */usr/lib/X11/scologin/Xconfig* and add the following line:

```
DisplayManager.display.authorize: True
```

For further details regarding the Xauthorization mechanism, see the **xauth**(X) manual page and Chapter 4, "Running remote programs" in the *Graphical Environment Guide*.

Network mail

Mail received via the SMTP protocol should not be considered genuine. It is quite easy to forge mail that, when received, looks like it came from any user on any system (for example *joe@sco.COM*), even though that user did not send the message.

If SMTP has been installed on the system, even mail that appears to have come from a local user may actually be a forged message. Only if the SMTP channel is removed from the mail configuration can mail be considered genuine. Even then, anyone with *root* privileges can create a forged mail message.

Therefore, mail headers should never be taken too seriously because they are so easily forged. Negotiations are under way for creating a more robust, secure mail system suitable for world-wide communications. Until a draft standard is written, all mail messages of vital importance should be cross-checked by some other means.

Administering a trusted system

You have already chosen the security scheme (Low, Traditional, Improved, or High) to be used on your system. Even if you did not choose to run a trusted (Improved or High) system, you should consider the following options, which are useful under any set of defaults:

- Assigning administrative roles and system privileges (this page) — deciding who will administer your system
- Controlling system access (page 232) — deciding how strict you want to be
- Logging out idle users (non-graphical sessions only) (page 233)
- Using auditing on your system (page 234) — deciding how and whether to use it

Assigning administrative roles and system privileges

The first basic choice you must make is who will maintain the trusted system. 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. Subsystem authorizations allow you to assign administrative roles rather than using a single *root* user to administer the system. Under the Low and Traditional security profiles, most subsystem authorizations (except **auth**) are assigned to users by default. To assign a subsystem authorization, see "Assigning subsystem authorizations" (page 32).

If you intend to operate a system that conforms to C2 requirements, you should grant subsystem authorizations based on the notion of "least-privilege": assigning subsystem authorizations based on their responsibilities. For example, the backups administrator is granted the **backup** authorization and the printer administrator is granted **lp** authorization. Only *root* should have all authorizations. Under this scheme, general users should be assigned as few subsystem authorizations as possible. Use secondary authorizations (page 34) to grant limited access to capabilities of a subsystem.

NOTE You might notice that each primary 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. 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.

subsystem(M) lists all the programs and data files associated with a subsystem. Most of the functions normally exercised by the superuser on non-trusted UNIX systems are delegated to the protected subsystems detailed in this section. However, some functions still need to be performed by the superuser. This includes mounting and unmounting filesystems, and traversing the entire file tree. Only the superuser can do everything. Restrict the *root* password to a few users and assign a responsible user to the *root* account.

Controlling system access

One important aspect of operation on a trusted system is preventing unauthorized access. The available restriction mechanisms fall into three categories, all of which can be customized:

- password restrictions (this page)
- terminal use restrictions (page 233)
- login restrictions (page 233)

You can also generate activity reports on each of these restrictions, such as the login activity for a terminal or group of terminals, or report on user accounts with passwords that are about to expire. See "Creating account and login activity reports" (page 241).

Password restrictions

The Department of Defense *Password Management Guideline* (also known as the *Green Book*) was used as a model for password restrictions. Users are subject to much stricter password checking than traditional UNIX systems. The system administrator can place restrictions on password selection and expiration.

Selection restrictions control whether users can pick their own passwords or have the system generate passwords for them. When chosen, the password can be subjected to simple or extensive checking for obviousness, again at the option of the administrator. These controls are described in "Controlling password selection" (page 24).

Password expiration restrictions determine how and when passwords expire — see "Controlling password expiration" (page 23). The lifetime of a password has three stages:

- *The password is valid.*
- *The password has expired.* (If permitted, the user can still log in and change the password.)
- *The password is dead.* (The user is locked out and the administrator must unlock the account.)

If users are not allowed to change passwords, new passwords must be assigned. Users must notify the administrator when their account is locked.

When a user logs in, the **prwarn**(C) utility warns them of impending expiration. There is no notification mechanism for the administrator other than use of regular reports on impending expirations (page 242).

A popular tactic among users on systems where periodic password changes are enforced is to change their password once, thus satisfying the requirement, then simply change their password back again to the one they used before. To prevent a user from doing this, 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 system-wide (*System Defaults* database) or per-user (*Protected Password* database).

By default, the user account initialization files (*.cshrc*, *.profile*, and so forth) call the **prwarn**(C) utility to warn users of impending password expiration and prevent their accounts from being locked. Expirations can be 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.

Terminal use restrictions

Terminals are gateways to the system. In addition to the use of account passwords, terminals can be protected from attempts to penetrate the system. You can define the maximum number of failed login attempts, high numbers of which are typically associated with attempts to crack an account password. Terminals that exceed the maximum permissible number of attempts will be locked and you, the accounts administrator, must unlock them. In addition, you can specify an interval that must elapse between login attempts, which can further thwart attempts to break a password. To change or examine terminal restrictions, refer to "Setting login restrictions on terminals" (page 30).

Login restrictions

As with terminals, user accounts have parameters associated with the number of login attempts and retry intervals. To change or examine login restrictions, refer to "Setting login restrictions on accounts" (page 29).

Logging out idle users (non-graphical sessions only)

A user who has not entered any commands or information for a long time may mean that the user has left the terminal and forgotten to log out. The **idleout**(ADM) command monitors line activity and logs out any user whose terminal remains idle longer than a specified period of time. You must be logged in as the superuser to run **idleout**.

WARNING **idleout** is not intended for use with the X server; **idleout** will log out active X processes.

To begin monitoring line activity for the system, enter:

idleout

The **IDLETIME** variable in the */etc/default/idleout* file determines how long a user's terminal can remain idle before the system logs the user out. If the value of **IDLETIME** contains a colon (:), **idleout** calculates the time in hours; otherwise, **idleout** calculates the time in minutes.

You can also specify the acceptable idle time on the command line in the one of the following forms:

idleout ***minutes***

or

idleout ***hours:minutes***

If you want **idleout** to run automatically when you reboot your system, add this line to the file */etc/rc2.d/P88USRDEFINE*:

```
idleout
```

Restricting root logins to a specific device

You can restrict *root* logins to a specific device by creating a special entry in the file */etc/default/login*. For example, to restrict *root* to the first console multiscreen you would add the following entry to */etc/default/login*:

```
CONSOLE=/dev/tty01
```

Using auditing on your system

Auditing keeps detailed records of system usage, enabling you to determine if any tampering has occurred (attempted or successful). However, auditing can require additional supervision and disk space, depending on how long it is enabled. Auditing is discussed extensively in Chapter 6, "Using the Audit Manager" (page 259).

NOTE It is not necessary to keep auditing enabled. It can be a useful tool if tampering is suspected. Because most systems calls are recorded when auditing is enabled, it is also an excellent program debugging tool.

Protecting the data on your system

The primary data protection on your system is the use of standard UNIX system permissions on files and directories. If you are unfamiliar with file permissions, see Chapter 3, "Directories and files" in the *Operating System Tutorial* and Chapter 7, "Protecting files and directories" in the *Operating System Tutorial*. Understanding the permission bits that you can set to protect files and directories is crucial to the security of your system. The default permissions for files created on your system are governed by the system-wide **umask**(C), which can also be customized by individual users.

SCO systems also include important filesystem features that extend the protection of 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 the system administrator. Other features are active, meaning that you can select them for particular objects. These active features include:

SUID/SGID bits and security

When the SUID (set user ID) or SGID bits are set on the permissions of a binary file, it executes with the UID or GID of the owner rather than that of the person executing it. An SUID/SGID binary has access to all the files, processes, and resources belonging to the owner or group of the binary file. This mechanism is used by the system to manage access to protected files. For example, **passwd**(C) is an SUID binary that allows users to change their password stored in the *Protected Password* database without allowing them direct access to this information. But SUID/SGID bits can be misused. Ordinary users should not be able to set these bits, and their use is restricted by the **chmodsugid** privilege (page 35).

SUID, SGID, and sticky bit clearing on writes

SCO systems guarantee that the SUID, SGID and sticky bits are cleared on files that are written. This prevents users from substituting another program in the file to take advantage of its SUID or SGID bits (which they could not otherwise set).

NOTE The clearing of SUID and SGID bits can be disabled if desired. Refer to "Disabling C2 features" (page 254) for more information.

An SUID bit shows as an "s" in the permissions of a file. Example 5-1 (this page) demonstrates bit clearing twice (user input is in boldface).

Example 5-1 Bit clearing examples

```
$ 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
```

In the example, user *blf* (the **id**(C) utility was used to show the identity) first uses the **cat** utility to replace the contents of the file *myprogram*. The SUID bit is removed during this process. The second example demonstrates that the bit clearing is even done on files owned by the same user. When *blf* strips the file (removing the debugging information in a compiled binary file), the SUID bit is also removed. 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 these bits on user programs without fear that the user can switch programs in the same file.

NOTE SUID and SGID permission bits do not work on shell scripts.

The SUID, SGID, and sticky bits are not cleared on directories. The SUID bit has no meaning for directories, and both the SGID and sticky bits have a meaning for directories that warrant their remaining there.

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 superuser may remove files from that directory. Other users are denied the right to remove files regardless of the directory permissions. Unlike with file sticky bits, the sticky bit on directories remains there until the directory owner or superuser explicitly removes the directory or changes the permissions.

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 for 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.

Example 5-2 (page 238) illustrates the power of such a scheme. The sticky bit is the " t " in the permissions for the directory.

Example 5-2 Sticky bit example

```
$ id
uid=76(slm) 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 slm      guru    19587 Mar 17 19:41 mine
-rw-------   1 slm      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.
$ 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
```

The only files removed are those owned by user *slm* (the user in the example). The user *slm* could not remove any other file, even the accessible file */tmp/openfile*. However, the mode setting of the file itself allowed *slm* 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***

To remove the bit, replace the " + " with a " - " in the **chmod** command.

Data encryption

Data encryption can also be used to enhance the security of your system through the **crypt**(C) command. These features are described in Chapter 9, "Using a secure system" in the *Operating System User's Guide*.

NOTE The data encryption software is not included with SCO systems, but is available by request only within the United States. You can request this software from your dealer or distributor.

Imported data

Files and filesystems brought into the system from elsewhere are a threat to the system if not handled properly. This section discusses techniques to use when importing files to your system.

Imported files

Do not take for granted the permissions on an imported file. Not only are the */etc/passwd* and */etc/group* files different on each system, but different security policies 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. The files are owned by the user importing the files. The **tar**(C) and **cpio**(C) programs only preserve the ownerships of files loaded from an archive when invoked by the superuser. The archive programs generally reset the file modes to those described on the media containing the archive. In addition to having a mode that is more permissive than necessary, files can have SUID, SGID (page 235) 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 **-itv** 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 move the files, after adjusting the ownership and modes according to your system policy.

Imported filesystems

Mounting filesystems (including floppy disk filesystems) that were created or handled elsewhere raises all the same concerns as importing files. Filesystems also bring additional concerns:

- read-write filesystems may be corrupted
- file permissions (including SUID/SGID bits) may not be acceptable for your system

In either case, mounting a corrupted filesystem can cause the system to crash, the data on the imported file system to be further corrupted, or other filesystems to suffer damage from side effects. This is why the **mount**(ADM) command is reserved for the superuser. Read/write filesystems should always be checked with **fsck**(ADM) before they are mounted. If the filesystem contains system files, the **integrity**(ADM) and **fixmog**(ADM) utilities should also be run after it is mounted.

Mounted filesystems can contain unacceptable file permissions. The superuser of the imported filesystem may have set ownerships, sticky bits, special files, SUID/SGID bits (page 235), and file tree compositions incompatible with your system policies.

Administrators should consider very carefully which filesystems users are allowed to mount with the **mnt**(C) command. It is possible to circumvent system security after mounting a filesystem created in a separate administrative domain. For this reason, users should not be allowed to mount filesystems from devices which are removable, such as floppies, CD-ROMs, removable hard drives, and so forth. This could be used to introduce an SUID root binary created elsewhere.

Use the **nosuid** option with the **mount**(ADM) command to ignore SUID bits on an NFS or CD-ROM filesystem. Only those NFS filesystems that are controlled by the same set of superusers should be excepted from this recommendation.

You can also use the **-s** option to **ncheck**(ADM) to locate any potentially dangerous SUID 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.

Terminal escape sequences

Many terminals and terminal emulation programs provide features that can be exploited to circumvent system security. These features are typically accessed by so-called "escape sequences," sequences of characters in output text that have special meaning to the terminal. Typical attacks reconfigure the terminal in unexpected ways, or cause input to be sent to the system as if typed by the user.

If you are using terminals that provide these features, and the features can be disabled, you should disable them when you do not need them, particularly when executing system maintenance functions.

See the sections on the terminal subsystem in the **subsystem**(M) manual page for more information about escape sequences.

Creating account and login activity reports

Use the **Reports Manager** located in the *System/Security* directory of the SCOadmin hierarchy to monitor possible security problems with several reports:

- Account password status (page 242)
- Account summary (page 242)
- Terminal access and lock status (page 243)
- Login and account lock summary (page 243)
- Terminal login status (page 243)
- Account database consistency (page 250)

To save a report to a file, or send it to a printer, select **Save to File** or **Print Report** from the **File** menu.

Many reports use the following abbreviations:

Dflt System default. If a value is set for an individual account, terminal, or group, an actual value is shown — otherwise **Dflt** is used to indicate that the system default is used. See "About default selections" (page 9) for more information.

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 yes or no.

Reporting password status

In the **Reports Manager**, select **Password Status** from the **Reports** menu.

The report selections correspond to the three stages of password lifetime:

- valid (but soon-to-expire)
- expired (when the user can still change it)
- dead (when the account is locked)

The "soon to expire" report lists accounts with passwords due to expire within a week. Although an impending expiration is not actually an error, this report allows you to identify users who wait until the last moment to change passwords.

NOTE The default account configuration files (*.cshrc, .profile, .kshrc,* and so forth) automatically execute the **prwarn**(C) utility at login time to warn users about impending password expiration.

See "Controlling password expiration" (page 23) for more information.

Creating an account summary

In the **Reports Manager**, select **Account Summary** from the **Reports** menu. Select whether you wish a summary for a a user, a group, or all users.

This report summarizes the contents of the *Protected Password* database, and includes the following information:

- Password parameters — settings of the password expiration and selection criteria, which are abbreviated:

  Min minimum days between changes

  Exp expiration time (days)

  Life lifetime (days)

  Rnd? user can run generator?

  Pck? user can choose own?

  Rst? checked for obviousness?

  Lck? is account locked?

  These controls are explained in "Controlling password expiration" (page 23) and "Controlling password selection" (page 24).

- Last password changes, both successful and failed.

- Last logins, successful, failed, and consecutive failures. See "Setting login restrictions on accounts" (page 29).
- Privileges. See "Changing system privileges" (page 35).

Reporting terminal access status

In the **Reports Manager**, select **Terminal Access Status** from the **Reports** menu. Select a summary for a single terminal, a range of terminals, or all terminals.

This report summarizes the contents of the *Terminal control* database, and lists any lock conditions, unsuccessful attempts to log in at the terminal, and the configured delay between login attempts. As the number or unsuccessful attempts approaches the maximum login tries for the terminal, you should determine the cause for the problem. Most terminals should show a low number of login attempts.

See "Setting login restrictions on terminals" (page 30) for more information.

Reporting user login activity

In the **Reports Manager**, select **User Login Status** from the **Reports** menu. Select a summary for a user, a range of users, a group, or all users.

This report lists last successful login, unsuccessful login, the number of failed attempts, and whether the account is locked. As the number or unsuccessful attempts approaches the maximum login tries for the account, you should determine the cause for the problem. Most accounts should show a low number of login attempts.

See "Setting login restrictions on accounts" (page 29) for more information.

Reporting terminal login activity

In the **Reports Manager**, select **Terminal Login Status** from the **Reports** menu. Select a summary for a single terminal, a range of terminals, or all terminals.

This report lists the dates of the last successful and failed (good and bad) logins, including the number of failed attempts. See "Setting login restrictions on terminals" (page 30) for more information.

Logging unsuccessful login attempts

You have the option of logging consecutive unsuccessful login attempts made on a tty in a single session. (A session is limited to five attempts — a new session is started after the defined delay between login sessions elapses.)

To start logging, create the file */usr/adm/loginlog*. The information is logged like this:

```
humbert:/dev/tty03:Thu Aug 11 13:15:47 1994
humbert:/dev/tty03:Thu Aug 11 13:15:53 1994
humbert:/dev/tty03:Thu Aug 11 13:15:58 1994
humbert:/dev/tty03:Thu Aug 11 13:16:04 1994
humbert:/dev/tty03:Thu Aug 11 13:16:10 1994
```

NOTE The *loginlog* file is independent of the login limits enforced on terminals and accounts.

See also:

- "Setting login restrictions on accounts" (page 29)
- "Setting login restrictions on terminals" (page 30)

Detecting system tampering

No system can be considered completely secure. System penetration can be invited by something as simple as someone using an obvious password or leaving their terminal logged in overnight. The system is designed to identify and authenticate users properly. In addition, access to security-related data on the system is based on subsystem authorizations. If users have the proper authorization, then they can use system programs to modify the security databases (for example, the audit administrator can change the audit configuration, and the accounts administrator can change passwords).

The system prevents unauthorized users from making such changes, but identification and authentication is a critical step in this protection. These mechanisms are circumvented when users gain access to accounts having greater authorization than their own. After setting up your system to minimize the possibility of tampering, the remaining task is to discover whether any tampering has taken place. Tampering can result from:

- Users obtaining the password of another user or gain access to another account (as when someone leaves their terminal logged in).
- Users with authorization abusing their privileges.
- A knowledgeable user gaining unsupervised physical access to the computer.

Stolen passwords

Each time a user logs in, the system displays the time and date of the last log-in. The most obvious evidence of a stolen password is when this last login is different from what the user remembers; secondary evidence is that a user's files have been altered. Warn users to note their last login and report any discrepancies immediately, including any instances where their files have been disturbed. Make certain that users follow the guidelines discussed in "Login security" and "Password security" in the *Operating System User's Guide*. These guidelines ensure that other users cannot guess passwords or otherwise obtain them.

The administrator should carefully consider which restrictions to place on passwords. One popular (and dangerous) practice is to have accounts without passwords. Although this feature is available, accounts without passwords are strongly discouraged. It is difficult to prevent damage or further penetration of the system once someone has logged on to an account. The identification and authentication procedure is the first line of defense against tampering.

Another weapon against stolen passwords is the interval between login attempts and the limits on unsuccessful login attempts for accounts and terminals. Although this can be annoying when a user makes a mistake in entering a password, it hinders an unauthorized user making repeated attempts to guess a password.

Other than reports from the users themselves, the principal method for detecting stolen passwords is to generate terminal and login reports — see "Creating account and login activity reports" (page 241). Look for:

- logins made at off hours, or at times when the user is not supposed to be on the system.
- accounts with multiple unsuccessful login attempts.
- terminals with multiple unsuccessful login attempts.

If any of these occur, you should suspect that someone is trying to gain access to your system. You should ensure that passwords are both changed regularly and made difficult to guess; this is the best assurance of password security.

Abuse of system privileges

If you have reason to believe a person with *root*-level authorizations is abusing their privileges, you should enable auditing for that user to determine if they are performing questionable actions. If the user has the *root* password or the **su** authorization, event "L. (Admin/Operator Actions)" will show the actions done that require high-level authorization.

Unsupervised physical access to the computer

The most basic security requirement is to prevent unsupervised physical access to the system itself. Although it is often necessary for users to access the console to operate the floppy disk or tape drives, it is dangerous to leave the computer hardware open to unsupervised access after hours. A knowledgeable user might be capable of disabling the system and penetrating the *root* account. This is the most serious breach of security.

When no users are on the system, such an occurrence can go unnoticed. This kind of tampering can only be detected by looking for:

- login reports for the *root* account.
- suspicious superuser actions in the audit trail.
- unexplained system reboots in the audit trail.

The lesson here is to place your computer system under lock and key and disallow off-hour access to anyone other than designated system administrators.

Dealing with filesystem and database corruption

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 system. Filesystem corruption is an infrequent occurrence, but can result in the removal of files that are critical to the continued operation of your system. This notion of system integrity is different from dealing with tampering, which is the deliberate action of a malicious user to alter or access data. This section explains the important security database files and how to recover them in the event of a system crash.

The authentication database files

Several database files store the characteristics of the system, 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 the **authcap**(F) manual page.

WARNING The *Authentication* database files are not meant to be edited by hand. The trusted system utilities and SCOadmin applications maintain and display the information contained in the databases. We do not recommend modification through any other means.

The *Audit* and *File Control* databases are independent databases. The other databases described here (the *Protected Password* database, the *Terminal Control* database, the *Subsystem* database, and the *Device Assignment* 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 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 filesystem 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 stores device pathnames relating to the same physical device. For example, */dev/tty1a* and */dev/tty1A* 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.

Protected Password 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 setting up this database properly, 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 Control gives access to the system through terminals. It 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 is actually a series of files (one per subsystem) that list users who are given special privilege either to be a subsystem administrator or to perform special functions within a protected subsystem. These files are another element of the *Authentication* database, which 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 helps maintain the integrity of the TCB. 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.

Checking the system after a crash

Several programs are used to maintain the integrity of the *Authentication* database, the system area of the filesystem, and the filesystem as a whole. When checking or fixing problems, the 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 these programs after a system crash or abnormality in this order:

1. Run a filesystem check (page 79) with **fsck**(ADM). This occurs automatically at reboot after a crash.
2. Check for the absence of critical files (page 249) with **tcbck**(ADM). This is automatic at reboot time.
3. Recheck the consistency of the *Authentication* database (page 249) with **authck**(ADM).
4. Verify system software as described in "Verifying software" in the *SCO OpenServer Handbook*. This will check and repair problems with file permissions, including the repair of broken symbolic links.
5. Generate an audit report (optional).

Using the override terminal

An override terminal exists for *root* in case the security databases become corrupted and all logins are disallowed. This is a special entry in the file */etc/default/login*. The entry identifies the tty to be used when doing an override login for *root*. The default entry (shown below) permits *root* to log in on */dev/tty01*, also known as the first multiscreen on the console. You can change this default to be another login device.

```
OVERRIDE=tty01
```

When the databases are compromised and *root* logs in on the override terminal, this message is displayed:

```
The security databases are corrupt.
However, login at terminal tty is allowed.
```

When the account is locked and *root* logs in on the override terminal, this message is displayed:

```
Account is disabled but console login is allowed.
```

The tty used should be physically secure; remember that normal terminal locks do not apply to the superuser account on this tty.

Automatic database checking and recovery: tcbck(ADM)

When the system is halted suddenly by power or hardware failures, some filesystem damage can occur. Such damage can result in the removal of security database files, or can leave these files in an interim state if they were being updated at the time of the system crash. Whenever a reboot occurs, the system runs a series of programs to check the status of the database files. When the system terminates abnormally and is rebooted, this check is performed after **fsck**(ADM) is run on the root filesystem, prior to entering multiuser mode. See "Checking the security databases" in the *SCO OpenServer Handbook.*

NOTE **tcbck**(ADM) executes several checking utilities and even repairs certain inconsistencies (via **authck**), but does not execute **integrity**(ADM) or **fixmog**(ADM).

Database consistency checking: authck(ADM) and addxusers(ADM)

The **authck**(ADM) program checks the consistency of the *Authentication* database. There are several command line options that restrict the scope of the checking. This example option checks all databases and automatically repairs any inconsistencies found:

authck -a -y

The **addxusers**(ADM) command has the **-e** and **-s** options for checking the consistency of the UNIX account database files: */etc/passwd* and */etc/group*.

You can use the **Reports Manager** to generate reports based on these commands.

Creating UNIX system and TCB account database reports

In the **Reports Manager**, select **Checks and Verifications** from the **Reports** menu.

You have these options:

Check consistency of account databases
This report uses the **authck**(ADM) command to check each database. When the **authck** command is executed from the command line, you are given the option of allowing **authck** to repair any inconsistencies.

Check validity of /etc/passwd and /etc/group files
This selection does the same checking that is done when a new user is created, executing the **addxusers**(ADM) command with the **-e** and **-s** options. Error messages and warning messages are generated; any error messages must be acted upon.

System file integrity checking: integrity(ADM)

The **integrity**(ADM) program compares the entries of the *File Control* database against the actual file permissions on the system. It does not alter permissions. To repair permissions, see "System file permission repair: fixmog(ADM)" (page 251).

NOTE The verify functions of the **Software Manager** described in "Verifying software" in the *SCO OpenServer Handbook* are more extensive than those described here.

If your system is configured with the Low or Traditional security defaults, permission problems reported by **integrity** have no effect on system operation.

You should run **integrity** as follows:

```
/tcb/bin/integrity -m -e > int.report
```

Print the file *int.report* and examine it. **integrity** reports files and directories that are missing or have incorrect permissions or ownership. Here are sample messages generated by **integrity**:

```
/etc/utmp (entry 83) is wrong.
        Owner is root, should be bin.
        Group is root, should be bin.
        Mode is 0644, should be 0664.
/usr/spool/lp (entry 233) is wrong.
        Group is bin, should be lp.
        Mode is 0755, should be 0070.
/etc/inittab (entry 71) is wrong.
        Type is d. should be r.
/usr/lib/mkuser/csh (wildcard entry 216) is wrong.
        Owner is bin, should be root.
        Mode is 0700, should be 0750.
```

The owner, group, and mode refer to the file permissions. The file types "d" and "r" refer to directory, and regular file, respectively. Missing files should be replaced by restoring them from backups. Permission and "type" problems can be fixed with the **fixmog** utility. 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.

NOTE Some files may be listed as missing in a correctly configured system, such as one of the pair */usr/lib/cron/at.allow* and */usr/lib/cron/at.deny*.

System file permission repair: fixmog(ADM)

The **fixmog** command attempts to correct inconsistencies found by **integrity**(ADM). **integrity** traverses the *File Control* database and compares each entry to the real file in the filesystem. Each file is checked to ensure it has the specified owner, group, access permissions and type. **fixmog** changes the owner, group and access permissions of files to match the *File Control* database. You should always use the **-i** (interactive) option to ensure that you can confirm any changes before they are made.

NOTE The verify function of the **Software Manager** performs more elaborate checks and can fix broken symbolic links; see "Verifying software" in the *SCO OpenServer Handbook* for more information.

Understanding how trusted features affect programs

This section discusses features that affect system system programs (including daemons), and lists examples of procedural and programming changes that must be made to add and run new daemons properly on a trusted system. These will ensure that the daemons are started with proper identity and privilege, encounter no surprises if the system acts differently due to security features, and handle boundary conditions and failure cases properly.

You must carefully consider how each daemon program can run with proper behavior and safety. You should carefully test the daemon in a controlled environment and observe that it acts properly before opening it up for general use. This leads to fewer security problems introduced into your system, and fewer surprises when users attempt to use the daemon and receive unexpected results.

These features are discussed:

LUID enforcement

LUID enforcement requires that all processes have an LUID. Daemon processes that are **setuid** require special consideration on a trusted system. The only exceptions to the LUID rule are the processes that stamp the identifier on processes, namely the **init**(M), **login**(M), and **cron**(C) programs. (Technically, **getty**(M) also lacks an LUID, but it does not run set user ID programs). All trusted utilities either stamp their own LUID (as **auditd**(ADM) does) or assume that their LUID was stamped before they run (as **lpsched**(ADM) does). The **setuid**(S) and **setgid**(S) system calls fail if the LUID is not set.

The **cron** daemon is a special case and is allowed to run without an LUID. To start special daemons like **cron**, another daemon process, **sdd**, and a special utility, **sd**(ADM), are used to start and restart them. If you need to create a daemon that runs without an LUID, refer to the **sd**(ADM) manual page for more information.

NOTE If LUID enforcement has been disabled, use of the **sd**(ADM) command is unnecessary. See "Disabling C2 features" (page 254).

As administrator, you must ensure that every newly introduced daemon is stamped with an LUID if it is started from the system startup files (*/etc/rc?.d/**).

The proper procedure is to set up the */etc/passwd* and */etc/group* files with the proper pseudo-user and group accounts, and the *Protected Password* entry for the account. If the daemon is to be run from a startup script, add a line to that script like the one below to run the program from **su**(C) so that the identity of the process is set properly. The procedure is the same as running daemons under a certain account using the traditional startup scripts. For example, the line printer daemon **lpsched** is started with the following line:

```
su lp -c /usr/lib/lpsched >/dev/null 2>&1
```

The trusted version of **su** program sets the LUID for a process if it has not already been set.

stopio(S) on devices

The system has a feature that makes it difficult to handle console output from a daemon, and you must plan daemon output accordingly. All terminal devices are subject to the trusted system call, **stopio**(S), which was added to enhance the identification and authentication subsystem to prevent login spoofing. When a user logs out, the **getty** that is respawned on that terminal line calls **stopio** with the terminal device name as an argument. Any processes holding that device open are killed (signal **SIGHUP**) if they try to write to the device again. Daemons that write to the console are subject to this signal if a logout occurs at the console between daemon start up and daemon output. Because most daemons ignore **SIGHUP**, their message output is simply lost. Therefore, you should redirect daemon output to a file or disabled terminal if it must be preserved (or redirect the output to the null device as in the above example).

NOTE The use of **stopio**(S) on devices can be disabled if desired. See "Disabling C2 features" (page 254).

Privileges

Processes in the operating system run with a set of kernel privileges that control the special rights a process has for certain restricted system actions. If the daemon must take an action that requires one of those privileges, that account must be set up properly so that those privileges are applied to the daemon process. Refer to "Changing system privileges" (page 35) for more information on kernel privileges. If a daemon executes other SUID programs, it must have the **execsuid** privilege. If the process creates files with the SUID bit, it must have the **chmodsugid** privilege. If it uses **chown** to alter ownership of files, it must have the **chown** privilege. Processes that are not installed with the TCB should not run with any of the audit privileges. Other privileges are for special situations, and should not be allowed to non-TCB daemons.

Sticky directories

Another feature that may affect daemons is sticky directories (page 237). If a directory's mode includes this permission bit only the owner of the file or *root* can remove the file from the directory. Daemons that manipulate temporary directories may behave improperly if files that they had assumed they could delete cannot be deleted.

You can handle this situation in one of two ways. First, remove the directory's sticky bit. This solves the daemon problem, but users must be cautioned about the security implications of using that directory for holding temporary files. The other solution is to modify the daemon and its corresponding helper program to agree on a new convention for file sharing. This second situation assumes that you have source code available and that you have the expertise and budget to modify the application.

Disabling C2 features

In addition to customizing security defaults, you can also selectively disable C2 features to ensure compatibility with utilities that expect traditional UNIX system behavior. (In the Low and Traditional defaults, most C2 features are disabled by default). The following key features can be switched on or off by changing the associated kernel parameter:

LUID enforcement

Under C2 requirements, every process must have a login user ID (LUID). This means that processes which set UIDs or GIDs, such as the printer scheduler (**lpsched**), must have an LUID set when started at system startup in */etc/rc2.d*. This can cause problems with **setuid** programs. When the security mode is set to a lesser mode (that is, not "High"), enforcement of login user ID (LUID) is relaxed and **setuid** programs do not require an LUID to run. This feature is enabled by default when the High security default is selected, but it can be enabled or disabled by modifying the **SECLUID** kernel parameter. A value of 0 disables the enforcement of LUID.

Clearing of SUID/SGID bits on write

Under C2 requirements, the set user ID (SUID or **setuid**) and set group ID (SUID or **setgid**) bits on files must be cleared (removed) when a file is written. This prevents someone from replacing the contents of a **setuid** binary, but this can cause problems with programs that do not expect this behavior. In the lower security defaults, SUID and SGID bits are not cleared when files are written. This feature is enabled by default when the High security default is selected, but it can be enabled or disabled by modifying the **SECCLEARID** kernel parameter. A value of 0 disables this feature.

stopio(S) on devices

The **stopio**(S) call is used under C2 to ensure that a device is not held open by another process after it is reallocated. This means that other processes attempting to access the same device are killed. In the lower security defaults, **stopio**(S) is not called. This feature is enabled by default when the High security default is selected, but it can be enabled or disabled by modifying the **SECSTOPIO** kernel parameter. A value of 0 disables this feature.

These parameters can be changed by invoking the **configure**(ADM) command and selecting category 8: "Security," and changing the parameter desired. The kernel must then be relinked and booted for the new behavior to take effect. See "Relinking the kernel" in the *SCO OpenServer Handbook*.

Troubleshooting system security

This section lists problems and error messages 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:

Account is disabled – see Account Administrator

This message means the account is locked for one of three reasons:

- You locked the account with the **Accounts Manager**. See "Locking or unlocking a user account" (page 31).
- The password lifetime for the account elapsed. The password for the account did not change before the password lifetime was over. To re-enable the account, you can either assign a new password as described in "Setting or changing a user password" (page 22) or unlock the account as described above (in which case the user is forced to set the password at log-in time). Advise users to change their passwords before the lifetime expires.
- A number of unsuccessful tries were made on the account, exceeding 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 lockout. It may be that the user is a poor typist or that or another person is trying to lock the account and knows this is a way to do it, or that a real attempt to penetrate 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.

Account is disabled but console login is allowed
Terminal is disabled but root login is allowed

These messages are associated with the superuser logging onto the override terminal. 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 superuser account does not prevent the superuser 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 lockout caused by unsuccessful login attempts on the system console is cleared automatically, but lockouts due to other reasons remain in effect. The console, in effect, is never locked out for the superuser.

Audit: filesystem 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, 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 **Disable** selection of the **Audit Manager** (page 262) 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 *Protected Subsystem* database, and the *System Defaults* database, 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 intentionally vague. 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 the *Database Events* event was enabled for the user that generated the message. Running **authck**(ADM) (page 249) should help you determine the problem.

Can't rewrite terminal control entry for tty Authentication error; see Account Administrator

The most likely reason for this problem is that the device entry in the */etc/auth/system/ttys* file is corrupted. The best solution is to remove the file and rebuild it as described in "Cannot access terminal control database entry" (this page).

Cannot access terminal control database entry

This message means that a terminal device file entry is missing from the *Terminal Control* database. To ensure that all terminal devices that appear in */dev* are recorded in the *Terminal Control* database (*/etc/auth/system/ttys*), log in as *root* and enter these commands:

```
ttyupd
/tcb/bin/ale /etc/auth/system/ttys pttyupd
```

The first command ensures that all tty device entries in */etc/inittab* have an entry in the *Terminal Control* database. The second command performs a similar function with respect to pseudo-ttys (for example, */dev/ttyp1*).

WARNING Do not attempt to edit the *Terminal Control* database by hand.

Cannot obtain database information on this terminal

This message indicates a problem with the */etc/auth/system/ttys* file. Following the procedure in "Cannot access terminal control database entry" (page 257) should solve the problem.

Login incorrect

The user entered an incorrect login name or login/dial-up password. If this happens repeatedly, you may need to alter the password to permit the user to log in again.

login: resource Authorization name file could not be allocated due to: cannot open;

The */etc/auth/system/authorize* file was corrupted or removed. Follow the procedure in "Restoring critical security database files" in the *SCO OpenServer Handbook*.

Terminal is disabled – see Account Administrator

The terminal is locked to all users. This is similar to account locking. Either an authentication administrator locked the terminal or a number of incorrect login tries (to one or more accounts) passed the threshold for that terminal. In both cases, determine what happened, then unlock the terminal. See "Locking or unlocking a terminal" (page 32).

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. For more information, see "Changing system privileges" (page 35).

Unable to remove files

If a user has write permission on a directory, but is unable to remove files from that directory, the sticky bit has been set for that directory. The sticky bit is a directory protection setting that allows only the owner of the file (or *root*) to remove files from that directory. See "The sticky bit and directories" (page 237) for more information.

Chapter 6
Using the Audit Manager

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.

Auditing permits the review of the collected data to examine patterns of access to "objects" (files) 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. Because 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.

NOTE Another useful aspect of auditing is in debugging programs. Because an audit session can log specific activities, you can enable auditing while running a troublesome program and find out exactly what it was doing.

See also:

- Understanding the audit subsystem (page 260)
- Collecting audit data (page 271)
- Managing audit files and directories (page 277)
- Generating audit reports (page 280)

Understanding the audit subsystem

The audit subsystem is the most complex component of the security mechanisms on your system. Several topics are important to understanding how auditing works and how to use it:

- Audit subsystem components (this page)
- Audit methodology (page 263)
- Guidelines for effective system auditing (page 267)

Audit subsystem components

The audit subsystem consists of five major components:

- Kernel audit mechanism (this page)
- Audit device driver (page 261)
- Audit compaction daemon (page 261)
- Audit Manager interface (page 262)
- Data reduction and analysis facility (page 263)

Although not actually part of the audit subsystem proper, there are also 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 has a corresponding 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 further classifies the system calls into specific security events. The kernel audit mechanism makes an internal call to the audit device driver to write a record to the audit trail.

For example, 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 that event. However, if 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 *DAC Denial* event for *blf* with object */unix*. Consequently, a system call can map to a number of event types, depending on the object accessed and/or the result of the call. It is 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 example, **getpid**(S) retrieves the process ID of 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.
- 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. (The audit device is described on the **audit**(HW) manual page.) However, the audit device can only be opened by processes having **configaudit** or **writeaudit** privileges. This limits access to the audit device only to trusted utilities such as the audit daemon and the audit administrator interfaces. The audit device can be written to by many processes at the same time. The device handles the merging of the records into the audit trail. The device can 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, the subsystem creates a new collection file and begins writing to it this is configurable — see "Adjusting audit performance parameters" (page 274). The audit trail could, therefore, be viewed as a continuously growing sequential file even though many collection files are used.

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. 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 200KB of audit data in an hour. The daemon 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% reduction in file space, which greatly reduces disk space used to store 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 file's 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 opens a new collection file when the current one reaches a specified size, the daemon can create multiple compaction files to avoid growing a single file too large to be manageable — this is also configurable; see "Adjusting audit performance parameters" (page 274). 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 can 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 that do not have the **writeaudit** privilege. Because these subsystems cannot access the audit device driver directly but can interface to the daemon in a trusted manner, the daemon handles the writing of the application audit record to the subsystem.

Audit Manager interface

The **Audit Manager** allows the administrator to handle setup and initialization, modify subsystem parameters, maintain the subsystem (backup, restore, and so on), and reduce both general and selective audit data. You can start the **Audit Manager** in any of these ways:

- Double-click on the **Audit Manager** icon in the the *System/Security* directory of *System Administration* toolshed on the Desktop.
- Start the SCOadmin launcher by entering **scoadmin** on the command line, then selecting **System ➪ Security ➪ Audit Manager.**
- Enter **scoadmin audit manager** on the command line (or abbreviate to **scoadmin au**).

For more information on using SCOadmin managers, see "Administering your system with SCOadmin" in the *SCO OpenServer Handbook.*

Data reduction and analysis facility

The audit subsystem also includes a data reduction and analysis facility to examine 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 can identify all of the compaction files needed to reduce an audit session. Because 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 reduce the data selectively. In addition, 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 ignored 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 privileges

There are four privileges associated with the audit subsystem:

- The **configaudit** kernel authorization allows the audit parameters for all users of the system to be set.
- The **writeaudit** kernel authorization allows specific information to be recorded in the audit trail.
- The **suspendaudit** kernel authorization prevents any auditing.
- The **audittrail** secondary subsystem authorization allows users to generate audit reports on their own activities. When a user is assigned this authorization, they can access the **Report** selections of the **Audit Manager**.

Audit record sources

The audit trail contains the security-related events for the system. Effective auditing tracks 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, when, where (from what terminal), and what was done. Login failures are impossible to audit at the

kernel level because 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.

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 example, 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 whether that event is to be audited. In addition, the mechanism may 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) (page 225) contains a number of trusted applications essential to providing a trusted 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 example, 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, trusted applications run with the **suspendaudit** privilege, suspending kernel system call auditing for that process and allowing them to open and write the audit device. Only a few trusted applications are permitted to do this. A user process should not run with **suspendaudit** privilege. The privilege 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** privilege 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 trusted 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** privilege 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 provides audit support for both successful and unsuccessful login attempts. When a user is successfully logged in, **login** stamps the user process with 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 together 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. 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 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. These events are summarized in Table 6-1 (page 266); the letters are simple identifiers used to refer to the event by the audit subsystem (see Table 6-2, "Audit event descriptions" (page 272) for a more detailed list).

Table 6-1 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

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 (this page). The audit subsystem also maintains a mask of event types for each process (this page) on the system.

System audit event mask

The system event mask is global to the audit subsystem. You can change it during 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 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". Each process on the system has a "process event mask" that 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.

For each audit event type, the user-specific event mask has one of three values:

- 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 is set to the third value for all audit events, which causes 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 more or less than the rest of the user population.

Guidelines for effective system auditing

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 trusted 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.

You should follow certain guidelines when using the audit subsystem. The subsystem is designed to offer flexible performance and reliability and to let you collect the audit data that you want. Audit record generation supports "preselection" of audit events, user IDs, and group IDs. Preselection is valuable if you want to concentrate on a specific user or group of users for some reason (for example, when particular users have a pattern of attempting access to files to which they are not permitted). Event types may also be used for preselection such as auditing only login and logoff events. Preselection also saves disk space because the amount of audit records written to the collection files by the subsystem is reduced. There is, however, a drawback to using preselection. If a system security violation occurred and that event or the user that perpetrated the event was not selected for audit, the action is not recorded.

For this reason, it is more conservative to not preselect the audit events and users or groups, but instead to perform full auditing so that all security-related events are recorded in the audit trail. The disadvantage of full auditing is that it consumes a great deal of disk space.

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. If event preselection 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 performance, reliability, and security goals.

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 with 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 file-system 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 because the events leading up to a system crash are the ones that are most interesting 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 that could be 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 to 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 can be determined by elapsed time or an accumulated data threshold. Again, the choice is yours.

Security goals

The final area critical to audit subsystem administration is determining what needs to be audited. Preselection options for record generation can be used to fine tune the audit trail to concentrate on an event or several events. For example, the system may be limited in use to a small group of people but left unattended at night. Several dial-in lines may be provided for after-hours work. You may only be concerned with accounting for who uses the system and when. In this case, preselection can 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 that is 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. 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 achieve.

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 audit 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.

WARNING If the audit subsystem is disabled while the system is running and later re-enabled, a new session is created. 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. You can avoid this problem by disabling auditing only by taking the system down. Refer to "Maintaining audit trail continuity" (page 274) for more information.

Administrative concerns

There are several areas of concern for the audit administrator:

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 failures

Most systems crash at some time. If a system crash occurs, there is 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, but this does not guarantee that data always makes it to the disk, especially if a disk failure caused the system crash.

It is not uncommon to find filesystem damage on audit trail files upon reboot. 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 filenames.

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 sessions should be archived and removed using the **Files** selection of the **Audit Manager** before auditing is reenabled.

Collecting audit data

These tasks are related to data collection:

- Choosing audit events (this page)
- Auditing individual users and groups (page 272)
- Displaying current audit statistics (page 273)
- Enabling and disabling auditing (page 273)
- Adjusting audit performance parameters (page 274)

Remember that usage of the audit subsystem has two stages, collection and reporting. Each stage involves selection of audit data. Once initiated, the subsystem collects data as directed by the audit masks until auditing is terminated, or the system is halted. The system maintains two types of masks: the system-wide mask that governs default auditing done on all users, and an individual user mask that can be defined for each user. The user mask overrides the system-wide mask.

Choosing audit events

In the Audit Manager (page 262), select **Events** ➪ **Modify.** Use the arrow keys to move between event types. Use ⟨Space⟩ to toggle between "Y" (yes, audit) and "N" (no, do not audit). The event types are explained in Table 6-2 (page 272).

This event mask can be modified and dynamically altered for the current audit session, and it can be written to the parameter file to take effect on future audit sessions.

Table 6-2 Audit event descriptions

Event type		Description
A	Startup/Shutdown	system startups (boots) and shutdowns
B	Login/Logoff	successful and unsuccessful login attempts
C	Process Create/Delete	creation and termination of processes
D	Make Object Available	file, message, semaphore opens and filesystem mounts
E	Map Object to Subject	program execution
F	Object Modification	file writes
G	Make Object Unavailable	file, message, semaphore closes and filesystem unmounts
H	Object Creation	file/message/semaphore creation
I	Object Deletion	file/message/semaphore deletion
J	DAC Changes	file, message, semaphore permission or ownership changes
K	DAC Denials	denied permissions
L	Admin/Operator Actions	system administrator and operator tasks
M	Insufficient Authorization	tasks that failed due to insufficient privileges
N	Resource Denials	missing files and insufficient memory
O	IPC Functions	sending signals and messages to processes
P	Process Modifications	effective identity or working directory changes
Q	Audit Subsystem Events	system auditing enable, disable, modification
R	Database Events	security data changes and integrity
S	Subsystem Events	use of protected subsystems
T	Use of Authorization	superuser-only actions

Auditing individual users and groups

In the Audit Manager (page 262), select **Collection ➪ IDs ➪ Modify.** Enter the names of users and/or groups. Use the ⟨Tab⟩ to move between the fields.

You also have the choice of causing your selections to take effect during the current session or the next session. You can select different users and groups can be done many times within the same session. If no users and groups are selected, all processes are subject to the system-wide audit mask.

To set the audit mask for an indivdual user (instead of the system-wide mask) use the **usermod**(ADM) command:

usermod -x "{auditFlags { *flags* }" *user*

For example, this command sets the mask for events A-D for user *mavrac*:

usermod -x "{auditFlags {1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0}}" mavrac

You may find it easier to use a template file instead of typing the flags on the command line. See "Using account templates" (page 11).

Displaying current audit statistics

In the Audit Manager (page 262), select **Collection** ⇨ **Statistics.** A list of statistics is displayed on your screen.

Example 6-1 Audit collection summary example

```
*** 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
Highest number of audit buffers used:       1
Total number of audit buffer sleeps:        0
```

Enabling and disabling auditing

In the Audit Manager (page 262), select **Enable** or **Disable** from the main menu to switch auditing on or off.

The **Enable** function uses the current audit parameter file to perform the subsystem initialization. **Disable** exits gracefully from auditing (all collection files are read by the daemon and compacted). The daemon then terminates, leaving only an audit session log file and the session compaction files.

Most subsystem parameters can be modified while auditing is running, so you do not need to disable audit for that purpose. 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 trail continuity

There is an important consideration involving LUIDs and audit sessions. Example 6-2, "Incomplete audit trail example" (this page) is a section of an audit report that shows a denied file access, but with a user ID of *root* and not the unauthorized user ID that actually tried to access the file (in this case to **touch** the file */a*).

Example 6-2 Incomplete audit trail example

```
Process ID: 227 (*INC*) Date/Time: Thu Dec 14 18:47:16 1989
Luid: root  Euid: root  Ruid: root  Egid: root  Rgid: root
Event type: Access denial
System call: Creat
Object:  /a
Result: Failed-EACCES (Access denial)
Security policy: discretionary
```

Note the (*INC*) next to the process ID. This indicates that the audit trail for this process is incomplete. It means that auditing was started *after* this user logged in, so there is no record of the LUID being set and the reduction program does not know what it is. The reduction program assigns a value of zero (*root*) to any unknown LUIDs. The audit session must include the login for the user being examined, or the audit subsystem does not have a record of the user ID. The only way to ensure this is to start auditing before users are allowed to log in. You should avoid starting an audit session while the system is already active.

Adjusting audit performance parameters

In the Audit Manager (page 262), select **Collection ➪ Parameters ➪ Modify.** Use ⟨Tab⟩ to move between parameters.

NOTE For a discussion of performance issues, see "Performance goals" (page 268).

You can alter these parameters:

Write to disk every [ ] bytes

Write to disk every [ ] seconds

These two parameters control the frequency with which audit data is written synchronously to the audit collection file from the internal audit buffers. Flushing can 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 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 every [ ] bytes

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 which can later be reduced. To maximize the effectiveness of the compaction algorithm, the daemon needs to read blocks of data between 4KB and 5KB. This requires special handling by the subsystem because 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 be left at the default value of 4KB. Values greater than 4KB will not yield significant improvement.

Number of collection buffers

This specifies 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 because 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. At least two buffers are required. Most systems cannot effectively use more than 4-6 buffers to avoid performance problems. There is no simple way to calculate the optimum number of buffers. Generally, base this value on the expected process load of the system.

Collection file switch every [ ] bytes

Audit output file switch every [ ] bytes

These two parameters 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. Because 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 are 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 50KB, and the compaction files are 1MB. Make sure that the maximum size chosen for the compaction files does not exceed the **ulimit** established for the system, which determines the maximum size of a user file.

Compacted output files

This option is provided in case you want non-compacted audit files. There is no compelling reason why this option should be selected 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

This option starts auditing automatically each time the system is rebooted. This field is only displayed with the **View** selection; it is set according to whether auditing was enabled or disabled. If auditing was disabled, then auditing is disabled at startup.

Shut down auditing on disk full

This option allows the system to shut down automatically if the system runs out of disk space, thus avoiding data corruption.

Change parameters for this session

Change parameters for future session

These options dynamically alter the current session and/or make the changes a permanent part of the audit parameter file for future sessions.

Managing audit files and directories

Audit data is divided into sessions, with a new session started each time auditing is stopped and started once again. Audit data is examinined or manipulated by session number.

These tasks are related to file management:

- Listing audit sessions (this page)
- Backing up audit files (this page)
- Restoring audit files (page 278)
- Deleting audit files (page 278)
- Monitoring disk space consumption (page 278)
- Maintaining collection directories (page 278)

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 which sessions are still on the system and let you remove sessions no longer wanted.

Listing audit sessions

In the Audit Manager (page 262), select **Files ➪ List.** You see a list of current audit sessions stored on the system.

A new session is opened each time auditing is stopped and started again.

Backing up audit files

In the Audit Manager (page 262), select **Files ➪ Backup.** Enter the session number you wish to back up; press ⟨F3⟩ for a list of sessions. Next, select the backup device and the volume size.

Because 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 and restore interface provides this capability.

NOTE Auditing consumes a great deal 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 is probably not necessary to use the audit backup selection. Again, it is important to remove the files to free disk space after they are backed up.

Restoring audit files

In the Audit Manager (page 262), select **Files ➪ Restore.** Insert the media containing the saved session files into the restore device, and specify the device name.

Deleting audit files

In the Audit Manager (page 262), select **Files ➪ Delete.** Enter the session number you wish to delete; press ⟨F3⟩ for a list of sessions.

Monitoring disk space consumption

In the Audit Manager (page 262), select **Collection ➪ Statistics.**

The total count of audit data written is the number of bytes currently stored on the system for the current session.

Maintaining collection directories

These tasks are related to directory maintenance:

- Listing collection directories (page 279)
- Creating a collection directory (page 279)
- Deleting a collection directory (page 279)
- Adding a collection directory entry (page 279)
- Removing a collection directory entry (page 280)

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 are read by the audit daemon. You do not need to keep track of the directories into which files are written because 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 competition for disk resources. Also, auditing 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.

Listing collection directories

In the Audit Manager (page 262), select **Collection ➪ Directories ➪ List.**

You see a list of the current audit directories.

Creating a collection directory

In the Audit Manager (page 262), select **Collection ➪ Directories ➪ Create.** Enter each filename as an absolute pathname. There is no limit on the number of directories you may specify.

You also have the option of adding the directory to the list of available directories used by the audit subsystem:

At End adds new directory at end of existing list.

Insert inserts new directory before an existing one.

No does not add this directory to the collection directories list.

If no directories are specified, the subsystem and the daemon create all files in the *root* filesystem using the reserved audit subsystem directory */tcb/audittmp* (the default configuration file setup). Directories are used sequentially as they are filled with data; this is why it is necessary to specify the position. When session files are backed up and removed from the audit directories, the system places new audit data in the first directory.

Deleting a collection directory

In the Audit Manager (page 262), select **Collection ➪ Directories ➪ Delete.** Enter the directory to be deleted. Press ⟨F3⟩ for a list.

Adding a collection directory entry

In the Audit Manager (page 262), select **Collection ➪ Directories ➪ Add.** Enter the directories as absolute pathnames.

You can also add an existing directory to the list used by the audit subsystem. Directories are used in the order listed. A new entry can either be inserted into the list or placed at the end. When you are asked to select the directory entry to be added to the list and specify the placement, select **At End** or **Insert**.

Removing a collection directory entry

In the Audit Manager (page 262), select **Collection ⇨ Directories ⇨ Remove.** Select the entry to be removed. Press ⟨F3⟩ for a list. This removes an audit directory entry from the list of available directories.

Generating audit reports

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 reduction function uses a file called a *report template* 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 report template each time. You can build and save report templates used frequently in data reduction. When the actual data reduction is needed, you can use the files already built.

These tasks are associated with audit reports:

- Creating or modifying a report template (this page)
- Viewing a report template (page 282)
- Listing report templates (page 282)
- Removing report templates (page 283)
- Running an audit report (page 283)
- Understanding audit reports (page 285)

Creating or modifying a report template

In the Audit Manager (page 262), select **Report ⇨ Create** or **Report ⇨ Modify.** Enter the name of the template file to be created or modified. Use ⟨F3⟩ to select from a list of templates; several are included on the system.

You can select based on five criteria: events, times, users, groups, and files. If **All** is chosen for a category, all events, times, and so forth are selected. If you choose **Select**, you are prompted to select the desired criteria:

Events The audit events can be selected or all collected. Events not selected cause those records to be omitted from the output. Depending on the template you selected, the events will have a "Y" or "N" in brackets. To toggle an event from yes to no, use ⟨Space⟩. When you are satisfied, press ⟨Enter⟩ to save your changes.

Times The start and stop times for collection. Press ⟨F3⟩ to get a calendar. 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 would concentrate the analysis on those records that are likely to reveal what has happened.

Users/Groups Both users and groups of users can be singled out for audit. You can highlight a user name and press ⟨Enter⟩, or use ⟨Space⟩ to mark multiple users and press ⟨Enter⟩ when the list is complete. 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. You can highlight a filename and press ⟨Enter⟩, or use the ⟨Space⟩ to mark multiple files and press ⟨Enter⟩ when the list is complete. 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 pathnames 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 example, time interval, user ID, and object name can be combined for a single session. If a record is within the specified time interval that 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. Similarly, 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 report template. If no users, groups, and objects are specified, only event type and time selection is performed.

Viewing a report template

In the Audit Manager (page 262), select **Report ➪ View.** Use the ⟨F3⟩ key to select a template to view.

You cannot make changes through this selection; if you wish to customize a template, select **Modify** instead.

Once a template is selected, the fields are filled in as in this example:

```
File name:  [ example              ]

Events:       A B C D E F G H I

Times: from   Fri Jan 24 12:48:00 PST 1992
         to   Sat Jan 25 07:30:00 PST 1992

Users:        user1 user2 user3 user4 user5

Groups:       group1 group2 group3

Files:        file1 file2 file3 file4 file5
```

The selections at the bottom of the screen are used to open windows to display data for each category that does not fit on the screen.

Listing report templates

In the Audit Manager (page 262), select **Report ➪ List.**

These report templates are shipped with the system:

admin.actions
: administrator actions; this is event type L.

all.objects
: actions relating to the creation, modification, or removal of objects; these are event types D—I.

authorization
: use of authorization; these are event types M and T.

dac.events
: DAC (Discretionary Access Control) changes or denials; these are event types J and K.

denials
: DAC denials; these are event types A—K, and N.

login.action
: a record of logins (successful and unsuccessful) and logouts; this is event type B.

Removing report templates

In the Audit Manager (page 262), select **Report ➪ Delete**, then enter the name of the report template to remove. Press ⟨F3⟩ for a list.

Running an audit report

In the Audit Manager (page 262), select **Report ➪ Generate.** Enter the session number or press ⟨F3⟩ for a list; do the same for the report template. You are then asked where to send the output, to the terminal, a file, or the printer. It is best to direct the output to a file.

When the report generation begins, note that it may take some time if the volume of data is high. For example, if your report template does not specify dates and times for beginning and ending selection, the entire audit session is reduced, which could consist of tens of megabytes of data.

Example report and template

Example 6-3 (page 284) is a sample audit report based on a template with these characteristics:

```
Events:  B K M T
Times:   Start: Fri Feb 2 19:00  Stop: Fri Feb 2 21:00
Users:   johnp
Groups:  None
Files:   All
```

The report template concentrates on undesirable activities, such as attempts to access restricted system files, running restricted administrative programs, and so forth. In this simplified example, user *johnp* logged on and attempted to remove (unlink) */etc/passwd*. In a real scenario, there would be more records to examine. This example serves to demonstrate the power of audit data. "Understanding audit reports" (page 285) contains a detailed study of how audit information is interpreted.

Example 6-3 Audit report output

```
                ***** Audit Data Reduction Program *****

        Audit session number:   2
        Collection system name: unix
        Collection file count:  15
        Compaction file count:  1
        Total audit records:    11034
        Total uncompacted size: 696050
        Total compacted size:   243262
        Data compression rate:  65.05
        Collection start time:  Fri Feb  7 19:00:15 1992
        Collection end time:    Fri Feb  7 21:00:00 1992

                ***** Selection Criteria *****

        Time Interval Selection:
                Start: Fri Feb  7 19:00:00 1992
                Stop:  Fri Feb  7 21:00:00 1992
        Event Type Selection:
                Event type: Login/Logoff activity
                Event type: Access denial
                Event type: Insufficient privilege

        UID selection in effect.

                johnp

                ***** Audit Records *****

Process ID: 235         Date/Time: Fri Feb  7 19:55:42 1992
Event type: Login/Logoff activity
Action: Successful login
Username: johnp
Login terminal:  /dev/tty01

Process ID: 267         Date/Time: Fri Feb  7 19:56:11 1992
Luid: johnp  Euid: johnp  Ruid: johnp  Egid: group  Rgid: group
Event type: Access denial
System call: Unlink
Object:  /etc/passwd
Result: Failed-EACCES (Access denial)
Security policy: discretionary

Process ID: 280         Date/Time: Fri Feb  7 19:58:14 1992
Event type: Login/Logoff activity
Action: Logoff
Username: johnp
Terminal:  /dev/tty01
```

Understanding audit reports

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 example, 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 facility 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 example, 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), **msgctl**(S), **shmctl**(S), and **semctl**(S) — that may only generate audit records for certain functions. For example, 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. Example 6-4 (page 286) illustrates the common header along with the system call and result fields.

Example 6-4 Common output record header

```
Process ID: 68          Date/Time: Sat Mar  7 13:25:09 1992
Luid: root  Euid: root  Ruid: root  Egid: root  Rgid: root
Event type:
System call:
Result:
```

Each system call is classified into a system event type based on the actions that are performed. This describes 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, **msgctl**(S) is the system call entry for message queue IPC operations. This single entry point calls **msgget**(S) and **msgop**(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 provides additional record classification. For example, 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 example, 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. The reduction associates this pathname with all further actions on that file, even though the action may have been performed with a file descriptor.

Table 6-3 System calls without pathnames

pipe	fork	kill	close
setuid	setgid	exit	security
read	setpgrp	msg	dup
sem	shm	write	fcntl

An output record from one of the above system calls uses the generic record mask described in Example 6-5, "setuid(S) system call record" (page 287), which illustrates the output record from a successful **setuid**(S) system call.

Example 6-5 setuid(S) system call record

```
Process ID: 6381        Date/Time: Tue Mar 17 11:25:19 1992
Luid: blf  Euid: blf  Ruid: root  Egid: root  Rgid: root
Event type: Modify process
System call: Setuid
Result: Successful
```

Similarly, Example 6-6 (this page) 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.

Example 6-6 Access denial output record

```
Process ID: 6381        Date/Time: Tue Mar 17 11:25:19 1992
Luid: blf  Euid: blf  Ruid: blf  Egid: guru  Rgid: guru
Event type: Modify process
System call: Setuid
Result: Failed (EPERM)-Not owner
```

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 **subsystems**(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 memory). **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 because it does not uniquely identify the file. Thus, the file descriptor correlates to a pathname and prints the pathname in the record.

Even though the **read**(S) and **write**(S) system calls are listed in Table 6-3 (page 286), 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 to track the state of the file. When the file is closed by **exec**(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 Example 6-7 (page 288).

Example 6-7 close(S) system call record

```
Process ID: 421         Date/Time: Sat Mar  7 17:15:09 1992
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
```

The second group of system calls, shown in Example 6-8 (this page), 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).

Example 6-8 System calls with pathnames

open	unlink	creat
exec	chdir	mknod
chown	chmod	stat
umount	exece	chroot
link	mount	

Each of the system calls in Example 6-8 (this page) 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 facility 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. Example 6-9 (this page) shows an output record generated from a **creat**(S) system call. The record format is the basic format augmented by the pathname.

Example 6-9 Output record with pathname

```
Process ID: 64         Date/Time: Sat Mar  7  23:25:09 1992
Luid: root  Euid: root  Ruid: root  Egid: root  Rgid: root
Event type: Object creation
System call: Creat
Object:  /tmp/daemon.out
Result: Successful
```

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 facility. A typical record produced by a **link**(S) system call is shown in Example 6-10 (page 289).

Example 6-10 Output record with two pathnames

```
Process ID: 14231      Date/Time: Thu Mar 18  03:25:39 1992
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
```

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. Example 6-11 (this page) illustrates the output record from a **chmod**(S) system call.

Example 6-11 chmod(S) system call record

```
Process ID: 6841       Date/Time: Sat Mar  7 13:25:09 1992
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
```

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 (page 290)
- user password events (page 290)
- protected database events (page 291)
- audit subsystem events (page 291)
- authorized subsystem events (page 292)
- terminal and user account lock events (page 292)

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.

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, and creates an important trail of user accesses and attempted accesses to the system. You can use the audit records for login or logoff to determine who actually used the system. It is also valuable in determining if repeated penetration attempts are being made. The system supports the option of locking terminals after a certain number of attempts and this event can also be audited. See "Setting login restrictions on terminals" (page 30) for more information.

Each login record indicates 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 login session. For unsuccessful attempts, the user name is meaningless, because the attempt failed. In this case, only the terminal on which the attempt occurred is output along with the basic record fields.

NOTE You can keep a record of unsuccessful attempts to log in (including the login name entered at the prompt) — see "Logging unsuccessful login attempts" (page 243).

Example 6-12 Successful login audit output record

```
Process ID: 2812      Date/Time: Fri Mar  6 10:31:14 1992
Event type: Login/Logoff Activity
Action: Successful login
Username: blf
Terminal: /dev/tty2
```

User password record

All attempts, successful or not, to modify a user account password are carefully audited by the authorization subsystem. To avoid revealing user passwords, 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.

Example 6-13 Unsuccessful password change audit record

```
Process ID: 7314      Date/Time: Tue Mar  3 18:30:44  1992
Event type: Authentication database activity
Action: Unsuccessful password change
Username: blf
```

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.

Example 6-14 Protected database output record

```
Process ID: 7314        Date/Time: Tue Mar  3 18:30:44 1992
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
```


Audit subsystem record

Events that affect the operation of the audit subsystem itself are audited very carefully. The **Audit Manager** 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:

- 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. Example 6-15 (this page) shows an actual audit record written to indicate the startup and initialization of the subsystem.

Example 6-15 Audit subsystem output record

```
Process ID: 517        Date/Time: Wed Mar  4  8:30:04 1992
Event type: Audit subsystem activity
Action: Audit enabled
```

Protected subsystem record

Each protected subsystem can generate audit records through the audit daemon. These records indicate unusual conditions detected by the subsystem. For example, 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 and availability of the system.

Aside from the normal record header output, the subsystem records contain a subsystem name, an action, and a result. The subsystem name is the subsystem that detected the inconsistency and wrote the audit record. The action and result describe the action taken by the subsystem and the problem detected.

Example 6-16 Authorized subsystem audit output record

```
Process ID: 2812      Date/Time: Fri Mar  6 10:31:14 1992
Event type: Authorized subsystem activity
Subsystem: System Administrator Subsystem
Security action: Update /etc/rc
Result: Cannot open for update
```

Terminal and 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 example, 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 log in to an account and fails repeatedly, that user account may be locked. Locking accounts and terminals prevents further access until the system administrator clears the lock. See "Setting login restrictions" (page 29) for more information. A terminal or user account lock may signal an attempted penetration of the system. These audit records contain the usual header information along with an identifier of the user account or terminal.

Example 6-17 User account lock output record

```
Process ID: 517      Date/Time: Wed Mar  4  8:30:04 1992
Event type: System administrator activity
Action: User account locked by system administrator
Username: root
```

Extending auditing capabilities to users

It is possible to extend some auditing functions to users. You can allow users to generate audit reports of their own activities. The **audittrail** secondary privilege permits access to a subset of audit functions under **Report** in the **Audit Manager**. Report output is limited to records matching the user's LUID. Users can use all report selections, including the creation of report templates, which are stored along with the system templates.

Refer to "Changing system privileges" (page 35) for information on assignment of the **audittrail** privilege.

Chapter 7

Connecting to other computers with UUCP

Use UUCP ("UNIX-to-UNIX Copy") to build a remote network system for your computer using normal telephone lines and modems. If your computer is on a local area network or is running TCP/IP, you probably do not need to set up UUCP. Local area networks and TCP/IP networks provide continuous connections that may be used in many ways, whereas UUCP provides only a scheduled file transfer capability.

NOTE UUCP is not a terminal emulation program. To log on to another computer over a modem, see Chapter 23, "Adding modems" in the *SCO OpenServer Handbook* and follow the instructions for connecting a modem.

The UUCP package consists of a group of programs that provide remote file transfer (**uucp**), remote command execution (**uux**), and mail to and from remote sites.

See also:

- "Setting up a simple UUCP connection" (page 296)
- "Configuring UUCP" (page 301)
- "How UUCP works" (page 320)
- "Advanced UUCP configuration" (page 331)
- "Troubleshooting UUCP" (page 336)

Setting up a simple UUCP connection

This section describes how to establish a modem-based UUCP connection to another computer. The connection described is a simple one-to-one connection between two systems using Hayes-compatible modems, or any modem specifically supported by a dialer binary (for example, Telebit, Multitech, or US Robotics). Both sites are able to dial into each other. It is assumed that the modem on each site is connected to */dev/tty1A*. (If your modem is attached to a different serial port, you will need to change this.) Finally, it is assumed that the modems are connected by a hardware-handshaking cables. For more detailed information and for other UUCP configurations, see "Configuring UUCP" (page 301).

To set up the standard UUCP system, you need:

- at least one RS-232 serial line (or serial port) on your computer to use for UUCP.
- the UUCP Utilities and Mail Utilities. If not installed, use the **Software Manager** or **custom**(ADM).
- a modem, configured as described in the Chapter 23, "Adding modems" in the *SCO OpenServer Handbook*.
- a standard telephone jack for access to the telephone system.
- a cable to connect the serial port to the modem.

To set up the connection, edit the */usr/lib/uucp/Systems* and */usr/lib/uucp/Devices* files. You can use the **UUCP Manager/uuinstall** utility (page 297) to help you do this.

The *Systems* file in */usr/lib/uucp* contains a list of the systems that UUCP knows about, and specifies the device and method used to contact each system. The *Devices* file identifies the usable devices, speed to use, and the serial ports that the devices are connected to.

To configure UUCP on a computer called **topaz**:

1. Make a note of the:
   - modem speed to use in making the connection.
   - name of your computer (called **topaz** in this example).
   - name of the other UUCP host (called **emerald** in this example).
   - serial line connected to the modem (*/dev/tty1A* in this example).
   - login and password to use when connecting to the remote machine (if you do not know these, the remote system's administrator should be able to tell you).

2. Log in as *root* and run the **UUCP Manager** located in the *Networks* directory of the SCOadmin hierarchy, or use the **uuinstall**(ADM) command line. You see this menu:

```
                    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   (/etc/inittab)
    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 q to quit :
```

3. Select **Display or update site name**. You will be given options to display or update the local site name. Ensure that the site name for your system is correct.

4. Return to the main menu (by pressing ⟨q⟩) and select **Display or update list of remote sites**, which edits the */usr/lib/uucp/Systems* file. You see a menu like this:

```
    Display or update list of remote sites (Systems)
    ================================================

    1. Display the Systems file
    2. Add a new site entry
    3. Delete a site entry
    4. Change a site entry

    Choose an option (1-4), or q to quit :
```

   Select **Add a new site entry**. **uuinstall** prompts you to enter information about the new site a line at a time:

```
Site name       : emerald
Schedule        : Any
Device type     : ACU
Speed range     : 2400
Phone number    : 0123456789
Expect login    : ogin:-@-ogin:-@-ogin:
Send login      : topaz
Expect login    : ssword:
Send login      : bbsuucp
```

   These fields are described in detail in "Adding entries for remote sites to the Systems file" (page 303).

5. Review your site entry, then type ⟨q⟩ to go back to the main menu. Now select **Display or update direct- or dial-out lines.**

   You should see a menu like this:

```
Display or update direct- or dial-out lines (Devices)
=====================================================

1. Display the Devices file
2. Add a new device entry
3. Delete a device entry
4. Change a device entry

Choose an option (1-4), or q to quit :
```

   Having defined a site to contact, it is necessary to configure the device used to contact the site. In the *Systems* file (step 4) you specified that the device ACU should be used. So it is necessary to add (or change) the device entry for ACU:

```
Device type     ? ACU
Tty line        ? tty1A
Dialer line     ? -
Speed range     ? 2400
Dialer          ? hayes2400
Switch token    ?
New entry is accepted :
```

   These fields are described in detail in "Specifying dial-up parameters with the Devices file" (page 309). (If you are not using a Hayes compatible modem, you may need to change the "Dialer" field, and optionally the "Speed range" field.) Use the dialer same dialer name that you used when you configured your modem. The dialers provided are described in "Using dialers" in the *SCO OpenServer Handbook.*

   After adding or updating the ACU device, you should return to the main menu.

6. If you do not want to permit **emerald** to log in on your system, skip this step. Select **Display or update direct- or dial-in lines** to update */etc/inittab* to permit UUCP to receive logins on the appropriate device (*/dev/tty1A*). You should see a menu like this:

```
Display or update direct- or dial-in lines (/etc/inittab)
=========================================================

1.      Display inittab file
2.      Enable login on a line
3.      Disable login on a line
```

Select **Enable login on a line** to enable login on a line, and enter the name of the line. (Display the *inittab* file for a list of acceptable lines.) Note that if the kernel is subsequently relinked, it is necessary to reenable any lines allocated for UUCP logins.

7. Quit from **uuinstall**. Edit */usr/lib/uucp/Permissions* to specify the kind of access that your system (**topaz**) has to the other host (**emerald**) when dialing out, and the access **emerald** has to your system when dialing in. Add the following text:

```
LOGNAME=emerald                 \
MACHINE=emerald                 \
COMMANDS=rmail:rnews            \
READ=/usr/spool/uucppublic      \
WRITE=/usr/spool/uucppublic     \
SENDFILES=call                  \
REQUEST=yes                     \
```

   These fields are described in "Limiting access with the Permissions file" (page 314).

8. If you are permitting incoming calls, assign a login name and password for the machine calling you. For example, if **emerald** calls you, you should create an account for it. See "Creating login accounts for sites dialing in" (page 302) for details. You must tell the administrator of the other machine what the password is, so that they can configure their *Systems* file login script for your host (**topaz**).

9. To permit incoming calls, return to **uuinstall** and select **Display or update direct- or dial-in lines**. This lets you display, enable, or disable the devices on your system. You must enable the device used by ACU before UUCP can receive incoming calls.

10. Run **uuinstall** and select **Check consistency of UUCP files**. This will tell you if there are any incorrect permissions or detectable inconsistencies in your UUCP configuration.

11. To test the UUCP connection from **topaz** to **emerald**, select **Test connection with remote site**. **uuinstall** prompts you for the name of the site to contact, then runs a UUCP test session. See "Testing the UUCP connection" (page 300) for more details of testing a connection.

Testing the UUCP connection

To test your UUCP connection:

1. If you are configuring an outbound UUCP connection, **cu** must function in order for UUCP to work. Follow the instructions for configuring and testing the modem in Chapter 23, "Adding modems" in the *SCO OpenServer Handbook*.
2. If you are using a Hayes or compatible modem, make sure the volume on the modem is at an appropriate level. You must be able to hear the modem to carry out this test successfully.
3. 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 ***ttynn***.
4. Start the **uutry** program by entering:

   /usr/lib/uucp/uutry -r -x9 *sitename*

   NOTE The output from this test can be found in */tmp/**sitename***. If you need to call your provider for assistance, it is a good idea to save this output.

   See "How a UUCP transmission proceeds" (page 324) for a discussion of the output from **uutry**.
5. 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.
6. The dialer automatically disconnects any call that it cannot complete. To break out of the shell created by **uutry**, press ⟨Del⟩ or ⟨BREAK⟩. This returns control to the terminal while **uucico** continues to run, sending the output to a file in */tmp* with the name of the system called.
7. If the signal is not present, make certain:
   - the modem is connected to the telephone jack
   - the jack is connected to the phone system
   - the correct phone number is in the *Systems* file
8. 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
   - the correct tty line is in the *Devices* file
   - the modem's power is on
   - there are no *LCK..* files in */usr/spool/uucp*

Changing the system name

For instructions on altering the host or system name, see "Changing the system name" in the *Mail and Messaging Guide.*

Configuring UUCP

The example installation described in "Setting up a simple UUCP connection" (page 296) assumes the most simple kind of UUCP network in common use: a modem to modem connection between two peer systems. Once your connection is established you can customize the data transfer protocol, system contact information, access permissions, and modem parameters. Alternatively, other kinds of UUCP network (for example, UUCP over TCP/IP) can be configured.

For information on these procedures, see:

- "How UUCP works" (page 320)
- "Configuring UUCP over TCP/IP" in the *Networking Guide*
- "Adding entries for remote sites to the Systems file" (page 303)
- "Setting up polling" (page 302)
- "Limiting access with the Permissions file" (page 314)
- "Specifying dial-up parameters with the Devices file" (page 309)
- "Defining a communications protocol" (page 332)
- "Connecting two local systems using a direct wire" (page 335)

> **NOTE** If you are planning to route mail over your UUCP system, see "Running the MMDF Configuration Manager" in the *Mail and Messaging Guide* for instructions on configuring mail traffic to work over UUCP.

See "How UUCP works" (page 320) for a description of the functions of the UUCP configuration files mentioned in "Setting up a simple UUCP connection" (page 296). Read "How UUCP works" (page 320) carefully before running **uuinstall** so that you understand the UUCP database.

Setting up maintenance scripts

There are several aspects of system operation that are governed by shell scripts running as daemons. These scripts must be set up by the system administrator. See the **uudemon**(ADM) manual page for complete instructions.

- checking the UUCP directory for work (**uudemon.hour**)
- sending of status information to the UUCP administrator (**uudemon.admin**)
- cleaning of the UUCP spool directory (**uudemon.clean**)

Setting up polling

Dial-up UUCP systems are normally configured to exchange files automatically at regular intervals. Each time a scheduled exchange is due, the UUCP system polls the dial-in system to see if files are waiting to be transferred.

Use **uudemon.poll2** to set up the polling schedule. To run **uudemon.poll2**, you need an entry for the daily daemon and an entry for the hourly daemon in the */usr/spool/cron/crontabs/uucp* file as follows:

```
0 0 * * *       /usr/lib/uucp/uudemon.poll2 -d
0 * * * *       /usr/lib/uucp/uudemon.poll2
```

The **-d** flag refers to the daily daemon. The hourly daemon has no flags. The above example has the daemon run at midnight. You can change the time the daemon runs by altering the second field using a 24-hour clock.

To establish the hours and days that **uudemon.poll2** runs, create two files: */usr/lib/uucp/Poll.hour* and */usr/lib/uucp/Poll.day*. These files contain the systems to be polled and the times and days they are polled.

A sample *Poll.hour* file follows:

```
hanna   12 1 3
raven   2 6 10w
```

If the hour is followed by a "w", **uudemon.poll2** calls the site only if there is work to be done.

A sample *Poll.day* file follows:

```
hanna   1 3 6
raven   1 2 3 4 5
```

The days of the week are integers, with Sunday being 0.

Creating login accounts for sites dialing in

A dial-in site must provide a login entry for the sites that call it.

A UUCP login account is the same as an ordinary user account but it has a special login directory and login program instead of the normal user directory and shell.

NOTE "uucp" should not be used as the name of a UUCP user or login account. It is the name of the UUCP owner or administrator.

To create a UUCP login entry:

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. Use the following information to create the account:

   Login shell: uucp (*/usr/lib/uucp/uucico*)
   Home directory: */usr/spool/uucplogins/**username***

3. To create an account, see "Adding and modifying user accounts" (page 10).

   Passwords are optional, but recommended, for UUCP logins. For information about what happens when a UUCP system with no account tries to dial in, see "Preventing unknown sites from logging in" (page 333).

Disabling password expiration and locks for UUCP accounts

UUCP login accounts are created with no password expiration. To alter this, see "Controlling password expiration" (page 23).

NOTE Remember that UUCP login accounts are used by remote systems using a login script which cannot cope with a prompt for a new password. For this reason it is sensible to set up an infinite password expiration, with the password changed manually in consultation with the remote site using that UUCP login.

If you have difficulties with UUCP accounts being locked (messages like `dead account` are displayed), you can change the number of login attempts by selecting as described in "Locking or unlocking a user account" (page 31).

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 If you plan to route mail traffic over UUCP, you must also configure MMDF as described in "Running the MMDF Configuration Manager" in the *Mail and Messaging Guide.*

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 can 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 with the keyword "Never" as in this example:

```
guardian Never
```

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 (or range) of the device used in establishing the communications link.
phone	provides the phone number of the remote computer for automatic dialers. If you wish to create a portable *Systems* file that can be used at a number of sites where the dialing prefixes differ (for internal phone systems), refer to "Creating a portable UUCP Systems file" (page 333) .
login-script	contains login information (also known as a "chat script"). This is explained in "Creating login scripts" (page 308).

The schedule field

The ***schedule*** field 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 (Monday-Friday)
Any	for anytime
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. (This field is ignored when you set up polling with **uudemon.poll2**; see "Setting up polling" (page 302) for details.) The call must be initiated by the remote computer. In other words, your computer is in a passive mode with respect to the remote computer (see "Limiting access with the Permissions file" (page 314) for details).

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 6:00am and 8:00am.

For example, the following permits calls on Mondays, Wednesdays, and Fridays between the hours of 9:00am 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 5:00pm to 8:00pm, Monday through Thursday, and calls any time on Saturday and Sunday. This 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
```

By default, UUCP uses a method called exponential backoff to allow retry of failed calls. UUCP does not allow another call to go through until after the retry time has elapsed. This interval expands exponentially as the number of unsuccessful attempts increases. The ***retry*** field overrides the exponential backoff algorithm. If you set the retry field to 9, for example, UUCP allows another attempt to connect 9 minutes after each failure. The ***retry*** field cannot be set lower than 5 minutes.

UUCP does not automatically try a failed call again. You must have polling set up as described in "Setting up polling" (page 302) or manually invoke **uucico**(ADM). Any files not transferred due to a connection failure are transferred at the next successful connection to that system.

The device field

The ***device*** field selects the device type, in most cases an ACU (Automatic Call 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 - 1200 /usr/lib/uucp/dialHA12
```

Additional documentation on the ***device*** field is located in the */usr/lib/uucp/Systems* file supplied with your system.

The speed field

This field can contain a letter and speed (for example, **C1200**, **D1200**) to differentiate between classes of dialers — refer to "The speed field" (page 311). 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 intersect the ***speed*** field in the associated *Devices* file entry:

Systems:
```
gorgon Any ACU D2400-9600 14087672676 ogin:
 nuucp ssword: DontLook
```

Devices:
```
ACU tty1A - D2400-9600 /usr/lib/uucp/dialHA9600
```

If information is not required for this field, use a hyphen (–) as a place holder for the field.

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 (CD) 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 rate 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 highest common speed of the *Systems* and *Devices* baud rate ranges to the dialer when connecting. If the dialer connects outside of the baud range in the *Systems* file, it returns a bad baud rate error. Otherwise, it returns the baud rate of the connection.

The phone field

This field provides 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. (See "Creating a portable UUCP Systems file" (page 333) for details.) 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.

NOTE Most dialers treat the equal sign as a pause, so you may need to use more than one.

Do not use the comma (from the Hayes command set) in a *Systems* file entry when you wish to indicate a pause. Use hyphens instead.

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 do 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 opens communications between modems, and also recognizes and sends proper login and password sequences. The script 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 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 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 "Testing the UUCP connection" (page 300) 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.

Table 7-1 Login (Chat) script escape sequences

Character	Description
\N	sends a null character (ASCII NULL).
\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. (After this sequence is used, whenever a character is transmitted, the system waits for the character to be received before doing anything else.)
\e	turns echo check off.
\n	sends or expects a newline 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 whose value is the ASCII character represented by that number (for example: \007).
""	expects a null string.

Specifying dial-up parameters with 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 contains 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 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 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 a binary dialer program, a *Dialers* file entry, or **Direct** for a direct link device.

The type field

This field usually contains one of two keywords (**Direct** or **ACU**) the name of a Local Area Network switch, or a system name.

Direct indicates a direct link to another computer or a switch for **cu** connections.

ACU 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. You can add your own LAN switch entries to the *Dialers* file. If you are adding a network using TCP/IP you would use the special dialer types **TCP**, **TLI**, or **TLIS**. See "Configuring UUCP over TCP/IP" in the *Networking Guide*.

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 of the *Devices* file is matched against the third field of the *Systems* file entry:

Devices: `gorgon tty1a - 1200 direct`

Systems: `gorgon Any gorgon 1200 - ogin: nuucp ssword: DontLook`

You can designate a protocol to use for a device within this field. For more information, see "Defining a communications protocol" (page 332).

NOTE Although the use of a system name adds clarity, only defined tokens can be used.

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 the *Systems* file entries, for example:

Devices: `ACU tty1A - 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 1200bps. 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

This field has the following format:

dialer [***token dialer token*** ...]

For a direct line, this field contains simply the word **direct**, and no token is required.

For a simple connection to a dialer, this field contains the name of the dialer, and the token is omitted; by default it is taken from the phone number field of the *Systems* file entry.

For a dialer or a network dataswitch, this field contains the name of an entry found in the *Dialers* file (**develcon** and **micom** are examples of network data switches). Other dialer types are supported by binaries instead of *Dialers* entries. (Support for 801-type dialers is provided through use of separate lines for data and the dialer. See the *Devices* file for details.) UUCP recognizes a dialer as a binary if the name begins with a " / " or if there is an executable file by that name in */usr/lib/uucp*.

For more information on *Dialers* entries and binaries, see "Using dialers" in the *SCO OpenServer Handbook*.

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 matches 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. " \D " is implied; see Modems used with a local network switch (page 313) . 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 then make the connection to the other computer. In this type of entry, there is only one pair. The ***dialer*** portion matches a *Dialers* file entry, as shown in the following example:

Devices: `develcon tty13 - 1200 develcon \D`

Dialers: `develcon "" "" \pr\ps\c est:\007 \E\D\e \007`

Systems: `obie develcon ACU 1200 obie --ogin:-BREAK-ogin: nuucp ssword: mavra`

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 contains the token in the ***phone*** field, which 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 makes 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 modem is connected, the second pair is accessed, where **ventel** is the dialer and the token is retrieved from the *Systems* file.

Here 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 does 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).

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.

NOTE The following command provides a useful interpretation of the *Permissions* file:

/usr/lib/uucp/uucheck -v | more

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 No spaces are allowed within an option assignment. This means that any continuations in an option assignment cannot have spaces before the " \ " or at the start of the next line.

Comment lines begin with a number 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.

In this way it is possible not only to define permissions for sites calling your system, but permissions for when your site calls other machines.

Permissions file restrictions

When using the *Permissions* file to restrict the level of access granted to remote computers:

- A machine cannot have more than one **LOGNAME** entry.
- Any site that is called whose name does not appear in a **MACHINE** entry, has the following default permissions or 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 **LOGNAME** and **MACHINE** are often combined for convenience, but they function independently. For example, if a remote system logs in as *nuucp*, **uucico** will read the first entry containing **LOGNAME=nuucp** without regard to the **MACHINE** name.

Permissions options

This section describes each option, specifies how it is used, and lists its default values.

REQUEST

specifies whether the remote computer can request file transfers from your computer. When a remote computer requests a file, this request can be granted or denied. The following string specifies that the remote computer can transfer of files from your computer:

```
REQUEST=yes
```

The following string specifies that the remote computer cannot request files from your computer:

```
REQUEST=no
```

which is the default value (it is used if the **REQUEST** option is not specified). The **REQUEST** option can appear in either a **LOGNAME** (remote calls you) entry or a **MACHINE** (you call remote) entry.

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 ask if your computer has work queued for it.

The following string specifies that your computer can send the work that is queued for the remote computer as long as the remote computer is logged in as one of the names in the **LOGNAME** option:

```
SENDFILES=yes
```

This string is mandatory if your computer is in a passive mode with respect to the remote computer.

The following string specifies that files queued in your computer be sent only when your computer calls the remote computer:

```
SENDFILES=call
```

The call value is the default for the **SENDFILES** option. This option is only significant in **LOGNAME** entries because **MACHINE** entries apply when calls are made out to remote computers. If this option is used with a **MACHINE** entry, it is ignored.

READ and **WRITE**
specify the various parts of the filesystem that **uucico** can read from or write to. The **READ** and **WRITE** options can be used with either **MACHINE** or **LOGNAME** entries.

The default for both the **READ** and **WRITE** options is the *uucppublic* directory as shown in the following strings:

```
READ=/usr/spool/uucppublic
WRITE=/usr/spool/uucppublic
```

The following strings specify permission to access any file that can be read or written by UUCP.

```
READ=/
WRITE=/
```

The value of these entries is a colon-separated list of pathnames. The **READ** option is for requesting files, and the **WRITE** option for depositing files. One of the values must be the prefix of any full pathname of a file coming in or going out.

NOTE **READ** and **WRITE** options do not affect the actual permissions of a file or directory. For example, a directory with permissions of **700** only permits the owner to access it, and cannot be read or written by the UUCP user, no matter what access options are defined in the *Permissions* file. In addition to the proper **READ** and **WRITE** options, the paths must grant appropriate permissions to the UUCP user.

To grant permission to deposit files in */usr/tmp* as well as the public directory, the following values would be used with the **WRITE** option:

```
WRITE=/usr/spool/uucppublic:/usr/tmp
```

It should be pointed out that if the **READ** and **WRITE** options are used, all pathnames must be specified because the pathnames are not added to the default list. For instance, if the */usr/news* pathname was the only one specified in a **WRITE** option, permission to deposit files in the public directory would be denied.

You should be careful which directories you make accessible for reading and writing by remote systems. For example, you probably do not want remote computers to be able to write over your */etc/passwd* file so */etc* should not be open to writes.

NOREAD and **NOWRITE**

specify exceptions to the **READ** and **WRITE** options or defaults. The following strings would permit reading any file except those in the */etc* directory (and its subdirectories — remember, these are prefixes) and writing only to the default */usr/spool/uucppublic* directory:

```
READ=/
WRITE=/usr/spool/uucppublic
NOREAD=/etc
NOWRITE=/etc
```

NOWRITE works in the same manner as the **NOREAD** option. The **NOREAD** and **NOWRITE** options can be used in both **LOGNAME** and **MACHINE** entries.

CALLBACK

specifies in **LOGNAME** entries that no transaction takes place until the calling system is called back. There are two examples of when you would use **CALLBACK**. From a security standpoint, if you call back a machine you can be sure it is the machine it says it is. If you are doing long data transmissions, you can choose the machine that is billed for the longer call.

The following string specifies that your computer must call the remote computer back before any file transfers take place:

```
CALLBACK=yes
```

The default for the **CALLBACK** option is:

```
CALLBACK=no
```

The **CALLBACK** option is rarely used. If two sites have this option set for each other, a conversation never gets started.

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.

The **uux** program generates remote execution requests and queues them to be transferred to the remote computer. Files and a command are sent to the target computer for remote execution. Note that **COMMANDS** is not used in a **LOGNAME** entry; **COMMANDS** in **MACHINE** entries define command permissions whether you call the remote system or it calls you.

The default command that a remote computer can execute on your computer is:

```
COMMANDS=rmail
```

If a command string is used in a **MACHINE** entry, the default commands are overridden. For instance, the following entry overrides the **COMMAND** default so that the computers *owl, raven, hawk,* and *dove* can now execute **rmail, rnews,** and **lp** on your computer:

```
MACHINE=owl:raven:hawk:dove \
COMMANDS=rmail:rnews:lp
```

Full pathnames of commands can also be used. For example, the following command specifies that command **rmail** uses the default path:

```
COMMANDS=rmail:/usr/lbin/rnews:/usr/bin/lp
```

The default paths for your computer are */bin, /usr/bin,* and */usr/lbin.* When the remote machine specifies **rnews** or */usr/lbin/rnews* for the command to be executed, */usr/lbin/rnews* is executed regardless of the default path. Similarly, */usr/bin/lp* is the **lp** command that is executed.

Including the **ALL** value in the list means that any command from the remote computer specified in the entry is executed. If you use this value, you give the remote computer full access to your computer. Be careful, this allows far more access than normal users have.

The following string illustrates two points:

```
COMMANDS=/usr/local/bin/lc:ALL:/usr/bin/lp
```

- The **ALL** value can appear anywhere in the string. The pathnames specified for **lc** and **lp** are used (instead of the default) if the requested command does not contain the full pathnames for **lc** or **lp**.
- The **VALIDATE** option should be used with the **COMMANDS** option whenever potentially dangerous commands like **cat** and **uucp** are specified with the **COMMANDS** option. Any command that reads or writes files is potentially dangerous to local security when executed by the UUCP remote execution daemon (**uuxqt**).

VALIDATE

is used in conjunction with the **COMMANDS** option in **LOGNAME** entries when specifying commands that are potentially dangerous to your computer's security. It provides a certain degree of verification of the caller's identity. The use of the **VALIDATE** option requires that privileged computers have a unique login or password for UUCP transactions. An important aspect of this validation is that the login or password associated with this entry be protected. If an outsider gets that information, that particular **VALIDATE** option can no longer be considered secure. (**VALIDATE** is merely an added level of security to the **COMMANDS** option, though it is a more secure way to open command access than **ALL**.)

Careful consideration should be given to providing a remote computer with a privileged login and password for UUCP transactions. Giving a remote computer a special login and password with file access and remote execution capability is like giving anyone on that computer a normal login and password on your computer. Therefore, if you cannot trust someone on the remote computer, do not provide that computer with a privileged login and password.

The following **LOGNAME** entry specifies that if one of the remote computers that claims to be *eagle, owl,* or *hawk* logs in on your computer, it must have used the login *uucpfriend.*

```
LOGNAME=uucpfriend VALIDATE=eagle:owl:hawk
```

As can be seen, if an outsider gets the *uucpfriend* login or password, masquerading is trivial.

VALIDATE increases security by linking the **MACHINE** entry (and **COMMANDS** option) with a **LOGNAME** entry associated with a privileged login. This link is needed because the execution daemon is not running while the remote machine is logged in. In fact, it is an asynchronous process with no knowledge of what machine sent the execution request. How does your system know where the execution files came from?

Each remote computer has its own *spool* directory on your computer. These spool directories have write permission given only to UUCP programs. The execution files from the remote computer are put in its spool directory after being transferred to your computer. When the **uuxqt** daemon runs, it can use the spool directory name to find the **MACHINE** entry in the *Permissions* file and get the **COMMANDS** list. If the computer name does not appear in the *Permissions* file, the default list is used.

The following example shows the relationship between the **MACHINE** and **LOGNAME** entries:

```
MACHINE=eagle:owl:hawk REQUEST=yes \
COMMANDS=rmail:/usr/local/bin/lc \
READ=/  WRITE=/

LOGNAME=uucpz VALIDATE=eagle:owl:hawk \
REQUEST=yes SENDFILES=yes \
READ=/  WRITE=/
```

The **COMMANDS** option line shows that remote mail and */usr/local/bin/lc* can be executed by remote users.

In the **MACHINE** entry, you must make the assumption that when you want to call one of the computers listed, you are really calling *eagle, owl,* or *hawk.* Any files put into one of the *eagle, owl,* or *hawk* spool directories is put there by one of those computers. If a remote computer logs in and says that it is one of these three computers, its

execution files are also put in the privileged spool directory. You should validate that the computer has the privileged login *uucpz*.

Entries for **OTHER** systems

You may want to specify different option values for machines or logins that are not mentioned in specific **MACHINE** or **LOGNAME** entries. This may occur when there are many computers calling in that have the same set of permissions. The special name **OTHER** for the computer name can be used in a **MACHINE** or **LOGNAME** entry as follows:

```
MACHINE=OTHER \
COMMANDS=rmail:/usr/local/bin/lc

LOGNAME=OTHER \
REQUEST=yes SENDFILES=yes \
READ=/usr/spool/uucppublic \
WRITE=/usr/spool/uucppublic
```

All options that can be set for specific machines or logins can be used with the **OTHER** value, although the use of the **VALIDATE** option makes little sense.

Combining **MACHINE** and **LOGNAME** entries

It is possible to combine **MACHINE** and **LOGNAME** entries into a single entry where the common options are the same. For example, the following two entries share the same **REQUEST**, **READ**, and **WRITE** options:

```
MACHINE=eagle:owl:hawk REQUEST=yes \
  READ=/  WRITE=/

LOGNAME=uucpz REQUEST=yes SENDFILES=yes \
  READ=/  WRITE=/
```

These two entries can be merged as follows:

```
MACHINE=eagle:owl:hawk REQUEST=yes \
LOGNAME=uucpz SENDFILES=yes \
  READ=/  WRITE=/
```

How UUCP works

The UNIX system uses the HoneyDanBer implementation of UUCP. 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* and its subdirectories; these work and data files are described in "The UUCP spool directory contents" (page 352). The program **uucico** scans these directories 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). There are three directories associated with UUCP:

/usr/spool/uucp This is the working directory for UUCP. Work files, lock files, log files, and all UUCP communications traffic are stored here and in subdirectories.

/usr/spool/uucppublic
This is the publicly readable or writable target directory used for most file transfers. A tilde (~) can be used as an abbreviation for this directory in **uucp** commands (in **csh** it must be escaped: \~).

/usr/lib/uucp Most of the UUCP programs are stored here, as well as the supporting database or control files. The main user programs, including **uux** and **uucp**, are found in */usr/bin*.

The */usr/lib/uucp* directory also contains configuration files for UUCP (distinguished by their capitalized names). The most important to understand are:

Configuration contains information that establishes and configures the protocol used when transferring data to and from a remote computer. (This file is optional.)

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.

The **uucp** traffic is managed by three daemons (supervisory programs) which 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 performs remote program execution. **uuxqt** runs after the conversation between the **uucico** programs is completed. It searches the spool directory for execute files (*X.files*) 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.

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. 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. See "A sample UUCP transaction" (this page) for an example of a session.

When **uucico** logs in on the remote computer, the remote computer starts a **uucico** daemon. The two **uucico** daemons then negotiate the line protocol to be used in the file transfer. The local **uucico** daemon then transfers the file that you are sending to the remote computer. The remote **uucico** places the file in the specified pathname on the remote computer. After your local computer finishes sending files, the remote computer may take over and send queued files to your local computer.

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, **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 steps trace 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 **obie\!\~** would also work for the destination and the exclamation point need only be escaped (preceded by a " \ ") if the **csh** is used; the Bourne shell (**sh**) and Korn shell (**ksh**) do not require this.

2. A work file named *C.obieNxxxx* is created in the */usr/spool/uucp/obie* directory, 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.
7. **uucico** creates a lock file (*LCK..tty1a*) 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*, is said to be the "guest"; the called system, *obie*, is said be the "host". The host **uucico** checks the local *Permissions* file to confirm that the guest is authorized to transfer the file.
10. The guest (*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/kilgore* directory on the host (*obie*). 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 recorded the exchange in */usr/spool/uucp/.Log/uucp/kilgore*.
12. Unless the host system (*obie*) has requests of its own, a hangup request is sent, the connection is terminated, and the lock files are removed.

For remote command execution (using **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. This takes place after the modems hang up and **uucico** exits.

How a UUCP transmission proceeds

A UUCP session consists of four parts: connection, an initial handshake that establishes the protocol to use, a series of file transfers, and a final handshake. All four parts produce different characteristic debugging output, and it is possible to deduce which element of the session is failing by scrutinizing the output.

When the connection stage begins, **uucico** scans the *Systems* file for the system to connect to. Using the device field for the specified system, it opens the *Devices* file and connects to the named device.

A single example session is documented in the following sections.

Stage 1: Connection

The command to invoke this session is **uutry -x9 scolon**, which sets the debugging level to 9 and makes **uutry** attempt to connect to system *scolon*. The beginning of the output shows **uucico** reading the *Systems* and *Devices* files in an attempt to set up a connection:

```
01: mchFind called (scolon)
02: list (rmail) num = 1
03: name (scolon) not found; return FAIL
04: list (rmail) num = 1
05: name (OTHER) not found; return FAIL
06: _Request (FALSE), _Switch (TRUE), _CallBack (FALSE), _MyName (), _Commands rmail
07: chdir(/usr/spool/uucp/scolon)
08: conn(scolon)
09: Device Type ACU wanted
10: mlock tty1A succeeded
11: fixline(6, 2400)
12: processdev: calling setdevcfg(uucico, ACU)
13: gdial(hayes2400) called
```

Line 01 indicates that **uucico** is looking for *scolon* in *Systems*. The next six lines are internal debugging information from **uucico**: on line 08 it finds *scolon* and attempts to connect to it. From the *Systems* file **uucico** determines that it needs a device of type ACU. Checking the *Devices* file, **uucico** determines that *tty1A* is such a device (line 10) and takes it over (line 11), setting it to the speed stated in *Devices*.

Having obtained a suitable device, **uucico** invokes the dialer *hayes2400* to place the call (line 13), and enters the connection script:

```
14: expect: ("")
15: got it
16: sendthem (ATQ0E0T&D2&C1S0=0X4S2=043^M)
17: expect: (OK^M)
18: ^M^JOK^Mgot it
19: sendthem (ATDT9222681^M)
20: expect: (Speed)
21: ^J^M^JCONNECT 2400got it
22: getto ret 6
23: expect: ("")
24: got it
25: sendthem (^M^M)
26: expect: (ogin:)
27: le /usr/spool/uucppublic/info contains a roadmap of ^J^Mthe bulletin board
areas.^J^M^J^MWARNING: Unauthorized access will be prosecuted.^J^M^J^M^M^Jscolo
n!login:got it
28: sendthem (uusls^M)
29: expect: (ssword:)
30: ic/info contains a roadmap of ^J^Mthe bulletin board areas.^J^M^J^MWARNING:
Unauthorized access will be prosecuted.^J^M^J^M^M^Jscolon!login: uusls^M^JPass
word:got it
31: sendthem (bbsuucp^M)
32: ISTRIP cleared
```

First the dialer talks to the modem. The standard Hayes modem response to a successful command is to echo OK; line 16 shows the dialer sending an initialization string (specified in */usr/lib/uucp/Dialcodes*, or programmed into the dialer binary) to the modem. (A Hayes **ATDT** command causes the modem to tone-dial the following number.) When the modem establishes a connection, the dialer waits for the modem to return the connection speed: in this case the standard message `CONNECT 2400` indicating a 2400-baud link. (Faster modems may return `CONNECT 9600` or `CONNECT 14400`.)

At this point, the dialer follows the login script given in *Systems*:

```
scolon Any  ACU 2400-9600 0923 222681 "" \r ogin:-@-ogin:-@-ogin: uusls ssword:
bbsuucp
```

First, a return is sent. Then, **uucico** waits for the string "ogin". This is received, and **uucico** sends "uusls", the appropriate login. (The remote host prints the text of */etc/issue*, a message file used for remote logins. **uucico** ignores the message.) Finally, **uucico** sees the password prompt, sends a password, and clears the login sequence.

If you run a sample **uutry** session and encounter problems during this phase, then it is probably due to one of the following conditions:

- Incorrect *Systems* configuration file. The host you are trying to contact is not in the *Systems* file, or its device type corresponds to no known device in the *Devices* file, or the login sequence is wrong.
- Incorrect *Devices* configuration file. **uucico** cannot initialize a device suitable for connecting to the designated host.
- Incorrect *Dialers* configuration file. (This is not normally needed for simple UUCP installations.)
- Dialer timeout before connection. If you are using a high-speed modem (particularly one that supports multiple speeds and modes) it generates a series of tones to negotiate a connection at the highest possible protocol and baud rate. This may take so long that the dialer is not allowing sufficient time for the connection. The solution is to add dashes "—" to the end of the telephone number in *Systems*. A dash is interpreted as a pause of approximately two seconds; if your modem is taking four seconds to negotiate a connection, add two dashes to the end of the telephone number to prevent the dialer from timing out.
- Incorrect hardware configuration or noisy line. A noisy line or badly configured modem can result in junk being received by **uucico**. In this case, **uucico** normally hangs up and **uutry** may attempt to repeat the connection later.
- Incorrect serial line configuration. If **uucico** begins displaying garbage after obtaining a connection (for example, after receiving the login message from the remote host), then the serial line from the local host to the local modem is probably running at the wrong speed. A solution is to check the login script in *System*; if it contains `-BREAK-`, replace each instance of `-BREAK-` with `-@-`, this prevents the UUCP programs from cycling the serial port speed.

Stage 2: Initial handshake

Now that the dialer has established a connection, the **uucico** daemon on the local host starts to negotiate a protocol with the remote **uucico**.

The ^J^M character pairs appearing in the **uutry** output are carriage return and line feed pairs. In DOS, both characters are needed to indicate the end of a line: in the Macintosh system a "^M" indicates the end of a line: and in the UNIX system a "^J" indicates the end of a line. Many online services transmit both characters at the end of every line, in case a different type of system tries to connect and cannot recognize the local newline format. When reading the **uutry** output, a ^M^J character pair should therefore be taken as indicating a new line.

omsg in the **uucico** debugging output means that the local **uucico** outputs a message: imsg means that the local **uucico** reads a message.

uucico ignores the incoming text until it sees a "^P" (byte with octal value \020). All messages in the initial handshake begin with a "^P" and end with a null byte (represented in this listing as "^@").

The first packet is ^PShere=scolon^@. The Shere= message indicates the remote **uucico** is running on *scolon*.

Here is the protocol negotiation phase:

```
CB + "protocol"
33 : imsg >^M^JLast   successful login for uusls: Wed Aug 11 14:28:58 BST 1993
on tty8C^M^JLast unsuccessful login for uusls: Tue Aug 10 19:41:17 BST 199ik3
on tty8D^M^JSCO UNIX System V/386 Release 3.2^M^JCopyright (C) 1976-1989 UNIX
System Laboratories, Inc.^M^JCopyright (C) 1980-1989 Microsoft Corporation^M^J
Copyright (C) 1983-1996 The Santa Cruz Operation, Inc.^M^J All Rights Reserved
^M^Jscolon^M^J^PShere=scolon^@Login Successful: System=scolon
34 : omsg "Sjocksco -Q0 -x9"
35 : imsg >^PROK^@msg-ROK
36 :  Rmtname scolon, Role MASTER,  Ifn - 6, Loginuser - root
37 : rmesg - 'P' imsg >^PPgetxf^@got Pgetxf
38 : wmesg 'U'g
39 : omsg "Ug"
40 :    Protocol: g
41 :            g window size set to 7
42 :            g variable packets set to true
43 :            g packet size set to 64
44 :    Protocol: G
45 :            G window size set to 7
46 :            G packet size set to 512
```

After receiving the first packet, `^PShere=scolon^@`, the originating host replies with an outgoing message:

```
34 : omsg "Sjocksco -Q0 -x9"
```

The second message packet in any transaction is:

^PS***hostname*** -Q***seqnum*** -x***debug***^@

where ***hostname*** is the local hosts name, ***seqnum*** is the sequence number for this conversation, and ***debug*** is the debugging level in use.

The sequence number is stored at both sites, and is incremented each time a conversation takes place via UUCP. If they do not match up, it is likely that security has been broken (or a disk backup restored following a crash). This feature is not used by all UUCP implementations.

The third packet is an acknowledgement from the remote **uucico** (which sends ROK — see below.) A number of different responses are possible at this stage; these may indicate various error conditions:

ROK	The calling UUCP is accepted; proceed to protocol negotiation.
RLCK	The called UUCP already has a lock file for the calling UUCP; the two machines are already communicating (or a lock file has not been removed).
RCB	The called UUCP will call back. (Used to avoid imposters.)
RBADSEQ	The call sequence number is wrong.
RLOGIN	The calling UUCP is using the wrong login.
Ryou are unknown to me	The calling UUCP is not listed in the recipient's *Systems* file, and the receiving system is not set up to accept anonymous UUCP connections.

If the connection is accepted, the remote system then sends a packet like `^PPgetxf^@`. The "P" prefix indicates that the subsequent letters are the protocols supported by the remote system. In this case, the "g", "e", "t", "x", and "f" protocols are supported, in decreasing order of preference.

The local **uucico** reads */usr/lib/uucp/Configuration* to obtain its own protocol setup, and sends a packet in response:

```
38 : wmesg 'U'g
```

to indicate the protocol (in this case, "g") to be used. (**uucico** uses the first protocol it finds listed against the appropriate system or device in *Configuration* that it has in common with the protocols listed in the remote **uucico**'s protocol string.)

The protocol is then established, and the rest of the UUCP debugging session consists of transfers.

Stage 3: File transfers

The debugging output from the file transfer stage is heavily dependent on the protocol being used to control the transfers. See "Defining a communications protocol" (page 332) for a brief introduction to the protocols.

At all times during a session, one **uucico** daemon acts as a master, issuing commands, while the other is a slave, obeying orders. Initially, the calling UUCP is master, although the roles may change during the session. If a protocol error occurs while commands are being exchanged, both **uucico**'s switch immediately to the final handshake.

The following commands can be sent by a master:

H A request to hang up the connection. The slave responds with one of:

- **HY** The slave agrees to hang up.
- **HN** The slave disagrees. Master and slave exchange roles, and the new master may begin to issue commands.

R A request by the master to receive a file from the slave. (Additional arguments are possible.)

S A request by the master to send a file to the slave. (Additional arguments are possible.)

X A request by the master to execute **uucp** on the slave. This command requires additional arguments. It is used when forwarding a file through intermediate hosts.

In the listing below, line 59 indicates that the "g" protocol is started (lines 47 to 58 are the internal "g" protocol startup sequence). No files are queued for transfer on either host, so after a brief exchange the master issues an **HY** command on line 76 (which is seen being sent on line 79), and the UUCP session moves to the closing handshake.

```
47 :    send 77
48 :    pkgetpack: Connodata=1
49 :    rec h->cntl 73
50 :    send 61
51 :    state - [INIT code a] (1)
52 :    pkgetpack: Connodata=2
53 :    rec h->cntl 61
54 :    send 57
55 :    state - [INIT code a]&[INIT code b] (3)
56 :    pkgetpack: Connodata=3
57 :    rec h->cntl 53
58 :    state - [O.K.] (10)
59 :    Proto started g
60 :    *** TOP ***  -  role=1, setline - X
61 :    gtwvec: dir /usr/spool/uucp/scolon
62 :    wmesg 'H'
63 :    pkwrite: icount = 64
64 :    send 37777777610
65 :    send 64 byte packet.
66 :    rmesg - 'H' pkgetpack: Connodata=4
67 :    rec h->cntl 41
68 :    pkcntl: RR/RJ: Connodata=0
69 :    state - [O.K.] (10)
70 :    pkgetpack: Connodata=1
71 :    rec h->cntl 37777777611
72 :    send 41
73 :    got HY
74 :     PROCESS: msg - HY
75 :    HUP:
76 :    wmesg 'H'Y
77 :    pkwrite: icount = 64
78 :    send 37777777621
79 :    send 64 byte packet.
80 :    send 10
81 :    send 10
```

Stage 4: Closing handshake

The closing handshake is used to confirm that the UUCP session has completed successfully. The calling **uucico** sends a packet of six zeros, and the remote host responds with a packet of seven zeros. (Line 86 is just a series of trailing nulls appended to the final response packet.)

```
82 :    cntrl - 0
83 :    omsg "000000"
84 :    send 00 0,omsg "000000"
85 :    imsg >^P^I^H*"^I^P^B!l^R]HY^@omsg "000000"
86 :    imsg
>^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@
^@^@^@^@^@^@^@^@^@^@exit code 0
87 :    Conversation Complete: Status SUCCEEDED
88 :
89 :    TM_cnt: 0
```

Different protocol debugging outputs

Different protocols produce different debugging output during the file transfer phase. The output in the preceding example uses the "g" protocol. The "G" protocol produces largely similar output, while the non-error correcting protocols ("e", "f", "x") produce different output.

In general, the debugging output from **uucico** once a protocol has been negotiated consists of a list of packets sent, and any errors or resend attempts made. The "g" and "G" protocols are particularly verbose: the other protocols tend to be less informative.

Advanced UUCP configuration

This section contains several options that are used only in special circumstances and can be ignored in most cases:

Defining a communications protocol

You can define the protocol to use with each device. In most cases it is not needed since you can use the default or define the protocol with the particular system you are calling (For more information on defining the protocol, see the *Systems* file "type" field.) To specify the protocol, use the form ***type,protocol*** (for example, **ACU, g**).

Alternatively, you can set up a *Configuration* file for your system. This file (*/usr/lib/uucp/Configuration*) contains the protocol configuration information used for contacting a specific system, or for use with a given device. You can specify different protocols in order of preference, and **uucico** will negotiate with the daemon at the other end of the session for the first protocol they have in common. In addition, the 'g' and 'G' protocols have configurable options. These are discussed in detail in the **Configuration**(F) manual page.

Table 7-2 UUCP communications protocols

Protocol	Description
g	standard UUCP protocol for connection over serial lines and modems. Uses error correction, provides variable packet and window size support and dynamic packet size adjustment.
G	variant on " g " protocol used by SVR4 systems. Provides variable packet and window size support. (This is disabled on some old fashioned " g " protocol implementations.)
e	protocol for 8-bit error-free links (example: TCP, TLI, TLIS). No error correction.
t	protocol for 8-bit error-free links (example: TCP, TLI, TLIS). Checks received file size. This protocol is provided for compatibility with BSD-derived systems.
f	protocol for 7-bit only error-free links (for example, some X.25 PADs). Does a checksum on the entire file.
x	protocol for 8-bit X.25 error-free links. Does not work on some X.25 packet switched networks (see **t** protocol).

NOTE Changing the **uucico** window parameter for the "g" protocol can cause compatibility problems. Although the "g" protocol definition allows different window sizes, some implementations have a compiled-in size of 3 packets. If the local **uucico** is configured to use a different window size, the remote **uucico** daemon may die if it has a compiled-in limit. For example, it is best not to modify the default values if you plan on connecting to UUNET. See the **Configuration**(F) manual page for a full discussion of issues surrounding protocol packet and window sizes, and "How a UUCP transmission proceeds" (page 324) for a demonstration of the debugging output from a "g" protocol session.

Creating a portable UUCP Systems file

The *Dialcodes* file (*/usr/lib/uucp/Dialcodes*) contains the dial-code abbreviations that can be used in the "Phone" field of the *Systems* file. This feature is intended primarily for those who wish to create a standard *Systems* file for distribution among several sites that have different phone systems and area codes. As such, the *Dialcodes* file is probably not necessary for most sites. See the **dialcodes**(F) manual page for more information, and "How UUCP works" (page 320) for general background on how the *Dialcodes* file is used.

Specifying alternate UUCP configuration files

The */usr/lib/uucp/Sysfiles* file lets you assign different files to be used by **uucp** and **cu** as *Systems*, *Devices*, and *Dialers* files. This feature s useful for cases where UUCP and **cu** require different dialers. See the **sysfiles**(F) manual page for more information.

Preventing unknown sites from logging in

The script **remote.unknown** is executed when a site whose name does not appear in your *Systems* file dials into your system. It logs the conversation attempt and fails to make a connection. If you wish to allow such "unknown" systems to log in to your system, you can change the permissions of this file so it cannot execute and your system accepts any communication requests. To do so, enter the following commands while logged in as *root*:

```
cd /usr/lib/uucp
chmod 000 remote.unknown
```

Configuring UUCP for 7-bit systems

Most UUCP systems use 8 data bits, one stop bit, and no parity for communicating over serial links. However, some implementations use 7 data bits, one stop bit, and one parity bit. Parity may be even, odd, or ignored.

To permit outgoing UUCP connections to a 7-bit system, you must modify the *Systems* file entry for that system. Two special keywords are recognized: **PEVEN** and **PODD**. **PEVEN** means even parity is to be used; **PODD** means odd parity is to be used. If both are specified, no parity is used. For example:

7-bit even:

```
remote Any ACU 2400 4253502 PEVEN -\r\d-ogin:-\K\d-ogin:-\K\d-ogin: uusys
```

7-bit odd:

```
remote Any ACU 2400 4253502 PODD -\r\d-ogin:-\K\d-ogin:-\K\d-ogin: uusys
```

7-bit none:

```
remote Any ACU 2400 4253502 PEVEN PODD -\r\d-ogin:-\K\d-ogin:-\K\d-ogin: uusys
```

To receive incoming calls from a 7-bit system, specify an appropriate */etc/gettydefs* entry in the */etc/inittab* entry for the modem port. As no 7-bit configurations are provided, you will need to add one. Be sure to leave blank lines above and below the new entry. For full details of a *gettydefs* entry, see the **gettydefs**(F) manual page.

The */etc/gettydefs* file recognizes three special keywords:

PARENB	use even parity
PARODD	use odd parity
IGNPAR	ignore parity

For example, assuming an incoming call is expected on *tty1A*, the following changes should be made to */etc/inittab* and */etc/gettydefs*:

7-bit even:

/etc/inittab

```
Se1A:2:respawn:/etc/getty -t60 tty1A 24E
```

/etc/gettydefs

```
24E# B2400 HUPCL # B2400 CS7 PARENB HUPCL TAB3 ECHOE IXANY #\r\n@!login: #24E
```

7-bit odd:

/etc/inittab

```
Se1A:2:respawn:/etc/getty -t60 tty1A 240
```

/etc/gettydefs

```
240# B2400 HUPCL # B2400 CS7 PARODD HUPCL TAB3 ECHOE IXANY #\r\n@!login: #240
```

7-bit none:

/etc/inittab

```
Se1A:2:respawn:/etc/getty -t60 tty1A 2DN
```

/etc/gettydefs

```
24N# B2400 HUPCL # B2400 CS7 IGNPAR HUPCL TAB3 ECHOE IXANY #\r\n@!login: #24N
```

For more information about parity settings, see the **stty**(C) manual page.

Connecting two local systems using a direct wire

If you are using UUCP to connect remote machines, you can skip this section. To connect two computers with a direct wire, you need to choose a serial port on each machine, connect a serial wire (RS-232) between the two machines using the chosen serial ports, and edit the UUCP configuration files.

Choosing a serial port

On each machine, choose the RS-232 serial port (*/dev/ttynn*) you want to use. If there are no ports available, you must install a new serial line or make one available by removing any device connected to it. Before you remove a terminal, make sure no one is logged in on it.

Find the name of the device special file associated with the line. 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. Be sure and use the non-modem control port (for example, */dev/tty1a* instead of */dev/tty1A*).

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.

Connecting a serial cable

Connect two computers together using an RS-232 cable. The actual pin configurations sometimes vary between machines.

The cable should connect pins 2, 3, and 7 on one computer to the same pins on the second computer. Typically, the cable must be *nulled*, which means that pin 2 on one machine is connected to pin 3 on the other, and 3 to 2. Because the connections can vary, check the hardware manuals for each computer to determine the proper pin connections.

Editing the UUCP configuration files

You should edit the UUCP files as instructed in "Configuring UUCP" (page 301), except you should use the keyword **Direct** instead of **ACU** in the *Systems* files.

Testing a direct wire connection

The following steps will enable you to use the **cu** command on the local machine to log in to the remote machine.

1. Enable the port of the remote machine:

   enable /dev/tty*nn*

   where ***nn*** is the serial port you are using.

2. Disable the port of the local machine:

   disable /dev/tty*nn*

   where ***nn*** is the serial port you are using.

To connect to the remote machine from the local machine, type:

cu -x9 -ltty*nn* dir

where ***nn*** is the port of the main machine.

Press the ⟨Bksp⟩ key until the lines are synchronized and you get a login prompt. To log in, you must have an account on the remote machine.

If you do not get a login prompt, read the **cu -x9** output for any clues of what could be wrong. See "Common UUCP error messages" (page 344).

Troubleshooting UUCP

Before troubleshooting the UUCP system, make sure that the physical connection works. If you are connecting to UUCP with a modem, verify that the modem is installed and configured correctly. See "Troubleshooting modems" in the *SCO OpenServer Handbook*. If you are using a direct line, make sure that the computers are connected correctly — use the instructions in "Connecting two local systems using a direct wire" (page 335).

NOTE If you are configuring an outbound UUCP connection over a modem, **cu** must function for UUCP to work.

If the hardware is correctly configured but a session is repeatedly failing for no known reason, the first troubleshooting step should be to run **uutry** and save the output log. The log (stored in */tmp/**hostname***, where ***hostname*** is the remote host's name) contains a listing of the steps **uucico** went through, and may indicate where the failure occurred.

uucico produces 11 levels of debugging output, from 0 to 10. Level 9 is the most verbose standard mode, and is generated when **uucico** receives the command line option **-x9** (you can also pass this option to **uutry** when testing the **uucico** connection to a given site). Level 10 is identical to level 9, except that the output from the "g" or "G" protocols is written to a file in */tmp* for future reference.

See also:

- "Checking for a faulty ACU or modem" (page 337)
- "Errors when testing the connection with cu" (page 337)
- "Common "UUCP failed" messages" (page 339)

Checking for a faulty ACU or modem

The following are two methods for checking whether the ACU (Automatic Call Unit) or modems are working correctly:

- To display a list of queued requests, the time of the last request attempt, and the contact status, enter:

  uustat -q

- To use a specific line and print debugging information during the contact attempt, enter:

  cu -x9 -l*ttynn phone*

See also "Troubleshooting modems" in the *SCO OpenServer Handbook*.

Errors when testing the connection with cu

This section suggests hot to respond to the messages displayed when testing the connection with the **cu** command fails.

Connect failed: CANNOT ACCESS DEVICE

If the connection fails and the system displays the `CANNOT ACCESS DEVICE` message, check the permissions on the device file. For example, to check the device file for *tty1a*, enter:

l /dev/tty1a

The ownership and permissions settings should look like the following:

```
crw--w--w- 1 uucp  uucp     5, 0 Feb 14 12:00 /dev/tty1a
```

If, instead, the ownership and permissions for the device file look like this:

```
crw------- 1 bin   terminal 5, 0 Feb 14 12:00 /dev/tty1a
```

verify that the line for the port in the */etc/inittab* looks similar to the following:

```
t1A:23:respawn:/etc/getty -t60 tty1A 3
```

If it does not, edit */etc/inittab* and */etc/conf/init.d/sio* and change the lines for the appropriate port.

NOTE Because a new *inittab* file is created each time the kernel is relinked, you must also edit */etc/conf/init.d/sio* to retain the changes in *inittab*.

To make your changes to *inittab* effective immediately, enter:

telinit q

Connect failed: SYSTEM NOT IN Systems FILE

If the */usr/lib/uucp/Systems* file on your computer does not contain an entry for the system that you are trying to access, **cu** displays this message.

NOTE This error message is often caused by an illegal character in the phone number, leading **cu** to treat it as a system name rather than a phone number.

To display a list of all the systems that your system is connected to, enter:

uuname

The system may display this error message if you used the **-l** option to specify a serial port and you entered the wrong line number. Verify that the line is configured properly.

Connect failed, NO DEVICES AVAILABLE

If **cu** fails to connect and displays the `NO DEVICES AVAILABLE` message, check to see that the */usr/lib/uucp/Devices* file is set up correctly.

Verify that the line that corresponds to the device that you are using is uncommented. For example, the entry in *Devices* for a direct line using *tty1a* at 9600 baud looks like this:

```
Direct tty1a - 9600 direct
```

There should be no leading spaces. If the *Device* file looks correct, the remote line may be busy. Try again later.

Connected, but no login prompt

If **cu** displays the `Connected` message, but not the login prompt, the line for the remote system may be busy. To exit **cu**, enter ~. and press ⟨Enter⟩.

If everything appears to be working on your system but the login prompt for the remote machine does not appear, check with the remote system administrator to verify that the **getty** on the remote system is set up with the same communications parameters that you are using.

Connected, but screen displays garbage

If you connect to the remote system, but your screen displays garbage, see "Problems dialing out" in the *SCO OpenServer Handbook* for more information. You should confirm that the communication settings on your system match those of the remote system. If the condition persists, the connection may be bad. Exit **cu** by entering ~. and try again later.

Common "UUCP failed" messages

Errors that UUCP displays immediately after you enter the **uucp** command are generally syntax or permissions errors, and include either of the following messages:

```
uucp failed completely
uucp failed partially
```

Here are some common problems and the error messages that the system displays:

- UUCP displays the following message when the permissions are set so that **uucp** cannot access the source file:

  ```
  can't read file (filename)
  uucp failed partially
  ```

  Change the permissions on the source file to allow all users read permissions.

- The following message is displayed when you do not enter both a ***source*** and ***destination*** with the **uucp** command:

  ```
  usage uucp from ... to
  uucp failed completely
  ```

- If you enter a site name that is not in the */usr/lib/uucp/Systems* file, UUCP displays:

  ```
  bad system name: sitename
  uucp failed completely
  ```

  Use **uuname** to display a list of the sites to which your system is connected.

Checking the status of a UUCP request

If the system does not display an error message immediately after entering the **uucp** command, and the system prompt appears, the **uucp** request is queued. Use the **uustat** and **uulog** utilities to check the status of your **uucp** job.

Use the **uustat** command to display the status of the currently queued **uucp** requests or display the status of connections to other systems. For example, to display a list of all the queued jobs for the user *robertm*, enter:

uustat -u robertm

The **uustat** utility displays user status information in the following format:

```
couscousN266 01/26-15:43 S couscous robertm rmail edwarda
```

To list the status of the accessibility of all the remote machines, use the following command:

uustat -m

The following example shows the output from the **uustat** command:

```
scooter 01/26-15:43  CAN'T ACCESS DEVICE
disco   01/27-12:01  SUCCESSFUL
obie    01/25-05:12  CALL FAILED, JOB DELETED
```

For more information on the options to **uustat**, see the **uustat**(C) manual page.

The **uulog** command displays the status messages (most recent last) that the UUCP programs write to the files *uucp/sys*, *uux/sys*, *uucico/sys*, and *uuxqt/sys*, in the */usr/spool/uucp/.Log* directory (***sys*** is the name of the system that **uucp** is trying to access).

For example, to display status information about file transfers involving the system *scooter*, enter the following:

uulog -s scooter

The **uulog** utility displays information about the system in the following format:

```
uucp scooter (01/26-15:43:00, 304, 5) COPY (SUCCEEDED)
```

NOTE The **uulog** command displays all the status messages that **uucp** logs in the *.Log* files. The messages are not necessarily error messages.

Alarms in UUCP audit output, data is not transferring

When sending data, UUCP expects a certain packet size and checksum on the packet. If it does not receive a full packet in a certain amount of time, a timeout occurs, which generates an alarm. A problem can sometimes occur once UUCP has begun sending data, in that part way through the transfer, alarms are generated and UUCP attempts to resend the packets. There are a couple of possible reasons for this problem.

First, check that **mapchan**(M) is not being used on the line that **uucico** is using. If it is, enter:

mapchan -n tty*xx*

to turn mapping off for that port. If **mapchan** is not running on that port, there is probably some kind of line noise or modem problem that is corrupting the data being sent.

Alternatively, the problem may occur if you are using UUCP with MNP 5 or some other error checking protocol that is specific to the modems. Error checking on a modem also sends a certain packet size with a checksum. The problem is that the sending modem waits to get a certain amount of data before it sends the packet, which can cause problems with UUCP because it is also waiting for a packet. The solution is to not use error checking on the modem when using UUCP (UUCP has its own error checking).

Generating log reports on usage: uulog

The **uulog** program displays log information on UUCP usage according to the remote machine. All usage of the programs **uucp**, **uuto**, and **uux** are logged in special log files, one per machine.

uulog options

The **uulog** command has the following options:

-f*system* displays the last entry or entries of the ***system*** file transfer log.

-s*system* displays the ***system*** file transfer information.

-x*system* displays the **uuxqt** log file for the given system.

-*number* specifies the ***number*** of lines displayed by the **-f** option.

For example, to print the last 10 lines of *chicago*'s file-transfer log, you would enter:

uulog -fchicago -10

Special uulog files

During execution of the **uulog** program, the files from the following directories are examined:

*/usr/spool/uucp/.Log/uucico/**
directory used for log files by the **uucico** program

*/usr/spool/uucp/.Log/uuxqt/**
directory used for log files by the **uuxqt** program

Common UUCP log and status file messages

UUCP keeps track of activity in log and status files in the */usr/spool/uucp* directory. The UUCP programs **uucp, uux , uucico** , and **uuxqt** write status information for each system to files in the */usr/spool/uucp/.Log* directories. The **uucico** utility writes messages to the files in the */usr/spool/uucp/.Status* directory. These messages describe the status of the transfer request, such as **uucico** failures, completions, and the time until the next allowable call for each remote system with which you communicate.

The following sections suggest responses to common error messages that UUCP programs write to these files.

DEVICE LOCKED

The **uucico** creates a lock file named *LCK..**sitename*** for the remote system in the */usr/spool/uucp* directory. If a file for the system that you are trying to call exists, **uucico** assumes that the device is in use.

To use the device, enter the following command on the remote system, substituting the name of the remote system for ***sitename***:

rm /usr/spool/uucp/LCK.*sitename*

NOTE If this does not solve the problem, the modem could be asserting CD (carrier detect) improperly.

LOGIN FAILED

If this error message appears, one of several things may be wrong:

1. Check the */usr/lib/uucp/Systems* file on the local machine to verify that the information in this file, particularly the chat script, is current. Some information that may be out-of-date is the phone number, login, and password.
2. Verify that the modem setup on both ends is correct. Refer to "Troubleshooting modems" in the *SCO OpenServer Handbook* for more information.
3. Check the phone connection. Try a different phone line.

After each step, invoke the **uutry** command.

NO DEVICES AVAILABLE

If the system displays this message, one of several things may be wrong:

1. The system assumes that the modem is in use because there is a lock file for it. Enter the following from the local machine:

   rm /usr/spool/uucp/LCK*

2. There may be no valid device for the calling system to use. Verify that the device named in the */usr/lib/uucp/Systems* file corresponds to the entry in */usr/lib/uucp/Devices*.
3. If the message persists, reboot the system. Enter the **rm** command again, and then invoke the **uutry** command again.
4. If the message persists after rebooting the system, the modem may be configured incorrectly. Verify that you can use **cu** to call out. Refer to "Troubleshooting modems" in the *SCO OpenServer Handbook* for more information.

REMOTE DOES NOT KNOW ME

If the system displays this message, the */usr/lib/uucp/Systems* file on the machine that you are trying to call does not contain an entry for your machine. Check the *Systems* file on the remote machine.

REMOTE HAS a LCK FILE FOR ME

If your system displays this message, either the remote system is trying to call your system or there is a lock file for your system on the remote system.

1. Remove the lock file from the remote system by removing the appropriate lock file:

   rm /usr/spool/uucp/LCK.*sitename*

2. Invoke the **uutry** command again.

SYSTEM NOT IN Systems FILE

If the */usr/lib/uucp/Systems* file on your system does not contain an entry for the system that you are trying to access, this error message is displayed. Use the **uuname** command to verify that the system name is in the *Systems* file.

RETRY TIME NOT REACHED

When UUCP requests fail, retries are not executed immediately. After an attempt to contact a remote system, a status file remains in */usr/spool/uucp/.Status/**nodename*** (***nodename*** is the name of the remote system that you are trying to reach). This file contains information about the last request and does not allow another request until the minimum retry period (specified in the */usr/lib/uucp/Systems* file) is reached. If you try to use UUCP again, the system displays an error message like the following:

```
RETRY TIME NOT REACHED
```

To enable another call attempt immediately, remove the status file for the remote system from the */usr/spool/uucp/.Status* directory:

rm /usr/spool/uucp/.Status/*nodename*

CANNOT ACCESS FILE

If the system displays this message, it cannot access the calling device port. Check the permissions and ownership on the device file:

1 /dev/tty*xx*

Where ***xx*** is the tty number. The ownership and permissions settings should look like the following:

```
crw-r--r--  1 uucp  uucp  5,  0 Feb  14 12:00 /dev/ttyxx
```

See "Connect failed: CANNOT ACCESS DEVICE" (page 337) for information on changing the permissions.

WRONG TIME TO CALL

The */usr/lib/uucp/Systems* file may restrict outgoing calls at this time.

Common 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. Error messages are also stored in */usr/spool/uucp/.Log/uucico/**sysname***.

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 text. In most cases, these errors are the result of filesystem problems. Use **errno** (when present) to investigate the problem. If **errno** is present in a message, it is shown as " () " in this list.

Table 7-3 ASSERT error messages

Error message	Description or action
CAN'T OPEN	An **open**( ) or **fopen**( ) failed. Check for the presence of the file and incorrect permissions.
CAN'T WRITE	A **write**( ), **fwrite**( ), **fprint**( ), and so on failed. Check for the presence of the file and incorrect permissions.
CAN'T READ	A **read**( ), **fgets**( ), and so on failed. Check for the presence of the file and incorrect 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 incorrect permissions.
CAN'T CHMOD	A **chmod**( ) call failed. Check for the presence of the file and incorrect permissions.
CAN'T LINK	A **link**( ) call failed. Check for the presence of the file and incorrect permissions.
CAN'T CHDIR	A **chdir**( ) call failed. Check for the presence of the file and incorrect permissions.
CAN'T UNLINK	An **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.
No uucp server	A TCP/IP call is attempted, but there is no server for UUCP.

(Continued on next page)

Table 7-3 ASSERT error messages
(Continued)

Error message	Description or action
BAD UID	The uid cannot be found in the */etc/passwd* file. The filesystem is in trouble, or the */etc/passwd* file is inconsistent.
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 message.
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* or *Systems* file ("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.
TOO MANY LOCKS	There is an internal problem!
XMV ERROR	There is a problem with some file or directory. It is probably the spool directory, because the modes of the destinations were supposed to be checked before this process was attempted.

(Continued on next page)

Table 7-3 ASSERT error messages
(Continued)

Error message	Description or action
CAN'T FORK	An attempt to *fork*(S) and *exec*(S) failed. The current job should not be lost, but is attempted later (**uuxqt**). No action need be taken.

UUCP STATUS error messages

UUCP 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, and whether it was successful or not. A list of the most common error messages that can appear in these files follows.

Table 7-4 STATUS error messages

Error message	Description or action
OK	Self explanatory.
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 or password, wrong number, a very slow machine, or failure in getting through the *dialer-token* script.
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.

(Continued on next page)

Table 7-4 STATUS error messages
(Continued)

Error message	Description or action
BAD LOGIN/MACHINE COMBINATION	The machine called us with a login or 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 "Common UUCP error messages" (page 344).
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 opening 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 lock 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 lock file. If they have the new version of UUCP and they are not communicating with your system, then the process that has a lock file is hung. This can also be caused by incorrect permissions in the */usr/spool/uucp* path on the remote system, or cleared **uucico** SUID bit.
REMOTE DOES NOT KNOW ME	The remote machine does not have the node name of your system in its *Systems* file.

(Continued on next page)

Table 7-4 STATUS error messages
(Continued)

Error message	Description or action
REMOTE REJECT AFTER LOGIN	The login used by your system to log in does not agree with what the remote machine was expecting. Check the *Permissions* file on the remote system.
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, and so on.
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.

Checking UUCP files and permissions settings

UUCP does not work correctly if it cannot read or execute its files. Because virtually all of the UUCP files are writable only by the superuser, and many of them are also readable and executable only by *root* and **uucp**, you should log in as *root* to install and modify the UUCP system. When you have finished, verify that all of the UUCP files are owned by *uucp* and not *root*.

To check the permissions of the UUCP files, use the **Software Manager** or **custom**(ADM) as described in "Verifying software" in the *SCO OpenServer Handbook*. Select **Expand Fully** from the **View** menu and then select "UUCP Utilities" from the list of components. Then select **Verify Software** from the **Software** menu. This verifies the UUCP distribution files and fixes any incorrect permissions.

Use the **uucheck**(ADM) command to check for the presence of the required files and directories.

NOTE The *Systems* and *Permissions* files contain unencrypted passwords, and therefore should be readable only by *uucp* (and *root*).

Verifying that your site name is unique

Make sure that the first seven characters of your sitename name are unique. If your machine is connected directly to a machine with the same sitename, UUCP does not allow communication with that machine. If both your machine and another machine with the same name are not directly connected, but are on the same UUCP network, electronic mail may go to the wrong machine.

To check the sitename use the **uname**(C) command. To change the sitename, see "Changing the system name" in the *Mail and Messaging Guide*.

UUCP truncates system names to seven characters

By default, UUCP truncates system names to seven characters to ensure compatibility with all versions of UUCP. Although disabling this feature is not recommended, it can be done.

In the file */usr/lib/uucp/Permissions*, add the variable **MYNAME=*name*** to each entry in this file, where ***name*** is the actual system name that contains more than seven characters. Also, add a new entry in */usr/lib/uucp/Permissions* as follows:

```
MACHINE=OTHER \
READ=/usr/spool/uucppublic:/usr/tmp \
WRITE=/usr/spool/uucpppublic:/usr/tmp \
COMMANDS=rmail:rnews:uucp \
SENDFILES=yes REQUEST=yes \
MYNAME=name
```

where ***name*** is the actual system name.

The **MYNAME** variable supersedes the truncated name. This procedure should be done on every machine in the UUCP network that has a name more than seven characters long.

NOTE **MYNAME** affects only those systems calling into the system with a *Permissions* file that has been modified as described here. Calling out to a system with a system name longer than seven characters results in that system name being truncated to seven characters. This is only a problem if your system calls systems with names which are identical through the seventh character.

What to check if UUCP is abnormally slow

Because UUCP is a batching network, requests are not executed immediately. Therefore, when users report that UUCP does not work, the problem may be that the system is slow. If the system is slower than usual, check for the following problems:

1. Check the spool directories, */usr/spool/uucppublic* and */usr/spool/uucp* for an overload of old work files and remove them.
2. Use **ps**(C) to verify that there are not too many **uucico** processes going at once. The limit to the number of communications processes is specified in */usr/lib/uucp/Maxuuscheds* and */usr/lib/uucp/Maxuuxqts*. Check to see that the limit is not too high (the default is 1).
3. Make sure that the filesystem that contains the spool directory is not out of space. Refer to "Maintaining free space in filesystems" (page 85) for more information.
4. The */usr/lib/uucp/uudemon.clean* script performs general cleanup by removing old files that are trying unsuccessfully to execute. To change the frequency of the cleanings, edit the **find** statement in the *uudemon.clean* script.

What to do if UUCP works, but uux does not

If you can use **uucp** to transfer files between two systems, but you cannot use **uux**, there is a problem with the */usr/lib/uucp/Permissions* file. When you use the **uucp** utility, the remote system requires only the **LOGNAME** entry in *Permissions*; **uux** also requires the **MACHINE** entry.

To fix this problem, add the **MACHINE** entry with the name of the remote system, to *Permissions*. For example, if your local machine, *goanna* is set up to call *obie*, the entry for *goanna* in the *Permissions* file on *obie* should look like this:

```
LOGNAME=uugoanna MACHINE=goanna \
        COMMANDS=ALL \
        READ=/ \
        WRITE=/ \
        SENDFILES=yes REQUEST=yes
```

NOTE The permissions granted in the example above are very liberal and should only be used in closely coupled systems where there is no security risk.

UUCP troubleshooting utilities

There are several commands you can use to check for basic communications information.

Table 7-5 UUCP troubleshooting tools

Command	Description
uucheck	Checks for the presence of files and directories required by **uucp**. This command also checks the *Permissions* file for obvious errors.
uulog	Displays the contents of the log directories for specific hosts.
uuname	Lists the machines that you are set up to contact.
uustat	Displays the status of the currently queued **uucp** requests or connections to other systems.
uutry	Invokes **uucico** with debugging, saves the information to the file */tmp/machine*, and directs the last 10 lines of the output to the terminal. The **-x** option changes the debugging level (default is level 5).

The UUCP spool directory contents

The following describes 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 (that is, */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 are 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 by converting the last letter to lowercase (meaning non-modem control) to avoid possible conflicts if the same line is specified both as 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 on the local computer when work (file transfer or remote command execution) 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 *~/**filename***; "~" is shorthand for */usr/spool/uucppublic* and must be included if the full pathname is not used
- 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 in the spool directory on both the local and remote computers 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 subsequence number ***yyy*** which 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 on the remote computer 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:

- requesters 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

Chapter 8

Administering virtual disks

A virtual disk is a set of disks configured into a disk array to provide improved system performance and increased data reliability. The disk array is seen by the applications that use it as a single (virtual) disk. Consequently, applications do not need any modifications to make use of the performance improvements that come with virtual disks.

Manage virtual disks using the **Virtual Disk Manager** (page 373).

See also:

- "About virtual disks" (page 356)
- "Planning your system layout with virtual disks" (page 369)
- "Adding virtual disks" (page 374)
- "Modifying virtual disks" (page 385)
- "Deleting virtual disks" (page 386)
- "Creating filesystems on virtual disks" (page 386)
- "Converting filesystems to virtual disks" (page 387)
- "Tuning the performance of virtual disks" (page 388)
- "Troubleshooting virtual disks" (page 389)

About virtual disks

Virtual disks are used to manage data in a more flexible way on systems with multiple hard disks. They are particularly useful for improving the performance of large applications, such as databases, by distributing the data across multiple disks and speeding up disk I/O.

Units of virtual disk space look like real disk partitions to programs running on the system, but their characteristics can be changed dynamically using the **Virtual Disk Manager**.

The **Virtual Disk Manager** adds an additional level of software control to the allocation of data storage. Normally, when applications request some data from the filesystem, the kernel uses the filesystem to discover the disk blocks where the data is stored and returns the data directly. When a virtual disk is in use, the system reads and writes to a virtual disk driver, which in turn manages the physical allocation of data across several disks.

This has a number of advantages. A virtual disk can be assembled from a collection of small disks, or pieces of disks, so that rather than providing a set of small partitions they can be used to provide a single large contiguous disk space. Data can be duplicated ("mirrored") across drives, so that if one drive succumbs to a hardware failure, the system can continue to operate without interruption: by using a technique called "striping", data can be read from and written to disks in parallel, significantly improving I/O performance.

You can select from eight virtual disk types on your SCO OpenServer system; see "Virtual disk types" (page 359) for details. Each provides different disk characteristics, suitable for different types of applications and for different hardware configurations.

See also:

- "Disk arrays and data striping" (page 357)
- "Hot spares" (page 357)
- "Clusters" (page 358)
- "Redundancy and parity" (page 359)
- "RAID" (page 359)

Disk arrays and data striping

A disk array organizes multiple independent disks into one large, high performance disk. In addition, some array configurations write backup parity information at the same time. (This renders them fault-tolerant, in that they can continue working even if a disk fails while in use.) Data blocks are split up and written in parallel to the disks, which speeds access.

The length of time taken to execute a read or write on a disk is determined by the time taken for the data area on the disk surface to pass under the read/write heads of the drive. Reading or writing an 8KB block takes eight times as long as a 1KB block. However, if the 8KB block is written to a disk array with eight disks, the data is split into eight stripes of 1KB, which are written to individual disks in parallel. In this way, disk arrays achieve a higher data transfer rate than non-parallel drives.

In practice, the expected linear scaling of throughput from using multiple disks is not achieved. This is because of seek-time, on-board disk caches and parity generation.

Data striping also results in uniform load balancing across all the disks on a system, eliminating disk hot spots. These arise when one disk is saturated with I/O requests while the rest lie idle.

However, when multiple disks are organized into arrays, the potential for data loss from disk failures is higher because the probability of a disk failure occurring in a given period is higher. For example, for a disk drive with a rated mean time before failure of 100,000 hours, there is a 50% probability of the drive failing in that length of time. However, for a disk array of 10 such drives, there is a 50% probability that one of them will fail in 10,000 hours (just over a year of 24-hour operation).

Hot spares

"Hot spares" are pieces which are allocated to a virtual disk and are used if one of the other pieces starts to fail. If I/O to a piece starts to fail, the hot spare piece is brought online, and data from the failing piece is moved to it (or restored from redundancy check data stored on the other disks in the array). When the data transfer is complete the hot spare takes the place of the failed piece. This ensures that the system is kept operational even if a disk fails under use.

If a drive in a RAID 1 (mirror) or RAID 5 virtual disk fails, the data availability is reduced immediately. The virtual disk is not protected from additional drive failures until the failed component is repaired. If a disk failure occurs in a virtual disk with a hot spare, data is automatically recreated on the hot

spare. When the recreation is complete, the virtual disk operates at its original level of data availability.

If a hot spare is not configured and a disk fails in an array, you can reconfigure the failed piece so it resides on a functional disk.

In the event of a disk failure, there may be some I/O performance degradation while data recovery takes place. Performance returns to its original level after the hot spare has been brought online.

It might be useful to use a single disk drive for all your hot spare pieces.

NOTE The Virtual Disk software does not support hot swapping of disks. Hot swapping allows a physical disk to be removed from an array, and another disk to be inserted to replace it.

See also:

- "Application and filesystem requirements" (page 369)

Clusters

A cluster consists of a set of contiguous blocks of data written to a physical hard disk within a disk array.

When data is written to a disk array, it is split into clusters that are allocated to different physical drives. For example, Figure 8-2, Example striped array (RAID 0) (page 362), shows an array of five drives. If a file is written to this array, it is split consecutively into stripes containing five clusters each. The first cluster is written to the first drive, the second cluster to the second drive, and so on. In this way, the stripe is written across the entire disk array (from left to right across the disks in the diagram). The cluster size is configurable from 512 bytes to 64KB in increments of 512 bytes.

If the file spans many stripes, every fifth cluster is written to the same drive. For example, on the diagram, clusters 0, 5, and 10 are written to the first drive; clusters 1, 6, and 11 are written to the second drive, and so on.

In Figure 8-2, Example striped array (RAID 0) (page 362), if Disk 1 had a capacity of 100KB, a cluster size of 5KB would divide each disk into twenty 5KB clusters. For example, blocks 0, 5, 10, ... 95 are clusters on Disk 1. Disk 2 contains blocks 1, 6, 11, and so on.

RAID

RAID is an acronym for redundant array of inexpensive disks.

RAIDs are replacing single, large, expensive disks because of the need for large, shared, high-performance, network-based storage systems. Such systems read and write data in parallel, and do so faster than a single disk system. They are also more reliable: the disks store data redundantly so that if one of them fails, the lost information can be interpolated from the other drives. Finally, they are cheaper. High capacity drives typically cost more per unit of storage than older, lower capacity drives, and disk arrays can use the cheaper units to achieve performance equal or superior to the more expensive, larger drives.

Redundancy and parity

RAID level 1, 4, and 5 arrays generate parity information, which is stored separately from the data it refers to.

If a disk in the array fails, the parity information and the partial data on the remaining disks are used to reconstruct the data on the failed disk. If the failed disk is the disk used for parity storage (in RAID level 4), the data from all the good disks is used to regenerate the lost parity information while the subsystem continues normal operations.

Virtual disk types

Eight types of virtual disk are available:

- "Simple disk" (page 360)
- "Concatenated disk" (page 361)
- "Striped array (RAID 0)" (page 361)
- "Mirrored disk (RAID 1)" (page 363)
- "Block-interleaved undistributed parity array (RAID 4)" (page 363)
- "Block-interleaved distributed parity array (RAID 5)" (page 365)
- "Striped, mirrored array (RAID 10)" (page 366)
- "Striped array of arrays (RAID 53)" (page 367)

This table summarizes their storage characteristics:

Table 8-1 Available virtual disk characteristics

Rating	I/O performance	Resilience
Best	RAID 10	mirror
	stripe	RAID 10
		RAID 53
	RAID 5, 53	RAID 4, 5 (hot spare)
	RAID 4	RAID 4, 5 (no hot spare)
	mirror	
	concatenated	simple
Worst	simple	concatenated, stripe

Simple disk

The simple virtual disk configuration lets you define all your non-*root* file-system space as one virtual disk. You can define the pieces that make up the virtual disk by specifying the starting block "offset" and length of the piece (a piece is a group of contiguous blocks configured to a specific virtual disk). Simple virtual disks supersede physical disk divisions created using **divvy**(ADM). Simple virtual disks can also be configured to overlay existing physical divisions. See "Converting filesystems to virtual disks" (page 387).

By overlaying a simple virtual disk definition on a standard disk division, you can make it easier to migrate your data to different virtual disk types. You can do this without interrupting system operation.

Simple virtual disks, like other virtual disks, are defined in the virtual disk configuration file */etc/dktab*.

This example is of a simple virtual disk entry:

```
/dev/dsk/vdisk4 simple
    /dev/dsk/2s1 2035 10000
```

The first line indicates the name of the virtual disk device file, and the type of the disk (simple).

The second line indicates the disk partition, the starting block allocated to the virtual disk, and the number of blocks allocated to the virtual disk (note that the number of blocks is reported in 512-byte blocks). In the device notation, */dev/dsk/dsp*, ***d*** is the disk number and ***p*** is the partition number. Disk numbers start from zero and partition numbers start from 1. See "Creating additional virtual disk nodes" (page 383) for more information on virtual disk node names.

NOTE Simple virtual disks can be configured to span an entire partition on the physical disk. However, the minimum offset is the size of the partition's reserved area.

Concatenated disk

A concatenated virtual disk is formed by adding two or more disk pieces (that may be physical disk divisions, previously defined virtual disks, or a mixture of both). This type of disk allows you to create a virtual disk that may be larger than any single physical disk in the system. The size of the concatenated disk is the sum of the sizes of all its component parts.

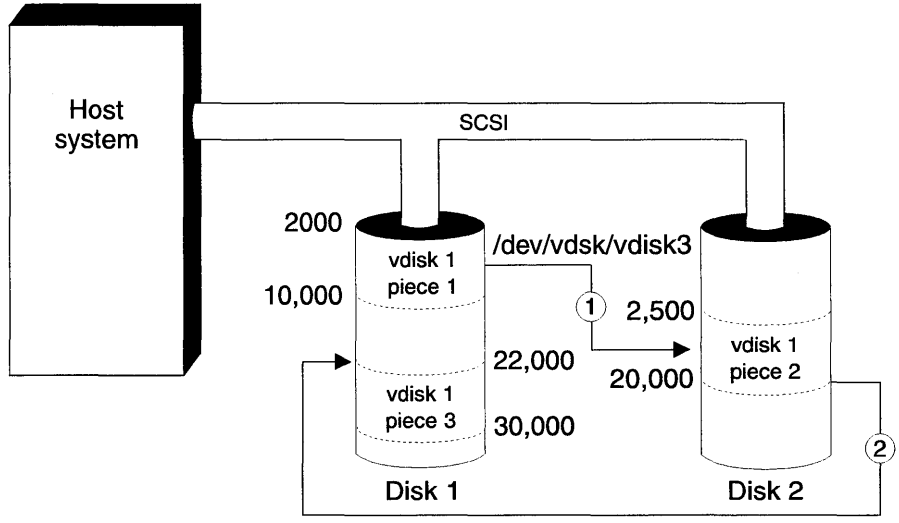


Figure 8-1 Example concatenated disk

The concatenated disk is referred to as */dev/dsk/vdisk4* and is made up of three virtual disk pieces. The */etc/dktab* file definition for the configuration is:

```
/dev/dsk/vdisk4 concat  3
    /dev/dsk/1s1        2000  8000
    /dev/dsk/2s1        2500  17500
    /dev/dsk/1s1        22000 8000
```

vdisk4 has a total capacity of 33,500 blocks (16,750KB).

Striped array (RAID 0)

Disk striping distributes data blocks across pieces stored on multiple disks. A "stripe" is a set of clusters written in parallel to a set of pieces on different disks; the "stripe width" is the number of bytes written in parallel. Because the pieces are written to in parallel, all the pieces must be the same size. A striped disk can consist of 2 to 255 of the same-sized disk pieces.

For information on optimizing application performance by varying the cluster size, see "Planning your system layout with virtual disks" (page 369).

This striped configuration, (RAID level 0), provides high write performance because there is no redundant information to update (redundancy is used to recover from lost or corrupt data). Without redundancy, any single disk failure will result in data loss. Non-redundant disk arrays are ideally suited to environments where performance and capacity, rather than reliability, are the primary concerns.

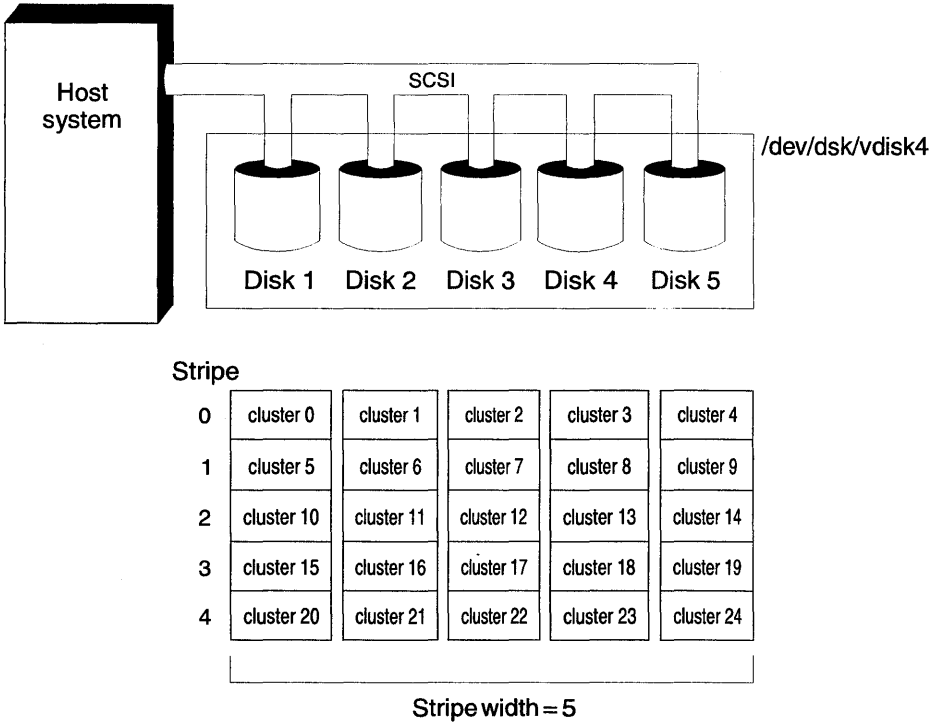


Figure 8-2 Example striped array (RAID 0)

The */etc/dktab* file definition for this configuration is:

```
/dev/dsk/vdisk5 stripe 5 8
    /dev/dsk/1s1 2000 497968
    /dev/dsk/2s1 2000 487968
    /dev/dsk/3s1 2000 497968
    /dev/dsk/4s1 2000 497968
    /dev/dsk/5s1 2000 497968
```

The first line indicates that */dev/dsk/vdisk5* is striped across five disks, with 8 blocks (of 512 bytes) per cluster. The second and subsequent lines indicate the disk pieces belonging to the virtual disk, with start block and end block numbers for each disk.

Mirrored disk (RAID 1)

Disk mirroring is the duplication of disk data onto a secondary disk. Both the primary and secondary disks are simultaneously online. Data written to the primary disk is simultaneously written to the secondary disk. Read requests are automatically directed to alternate disks to obtain the best performance.

When a mirrored disk is created, one disk is designated as the primary disk. Data from the primary disk is copied to the secondary disk until they mirror each other's content; thereafter, writes are directed to both disks in parallel.

Mirroring is ideally used in database applications where availability and transaction rate are more important than storage efficiency. Mirrors can also be configured across different buses.

An */etc/dktab* file definition for the configuration is:

```
/dev/dsk/vdisk4 mirror 2
    /dev/dsk/1s1 2000 798000
    /dev/dsk/2s1 2000 798000
/dev/dsk/vdisk5 mirror 2
    /dev/dsk/3s1 2000 798000
    /dev/dsk/4s1 2000 798000
```

Block-interleaved undistributed parity array (RAID 4)

RAID 4 is based on data striping (as in RAID 0 configurations) with additional redundancy information stored on a separate disk piece. Data is striped across three or more disk pieces using a defined cluster size. The added reliability of RAID level 4 is obtained by generating parity across the striped data; the parity is written to a separate disk piece.

Parity information is generated during disk writes and stored on an extra disk configured into the disk array for that purpose. In the event of any single disk failure, data is recreated for each block on the failed disk using the surviving data blocks and the parity block. The virtual disk will remain online, although performance will be reduced due to the need to recreate data from the failed disk. Except for some user warning messages on the console indicating the failure, the applications will not be affected.

Because a RAID 4 virtual disk has only one parity disk, and because that disk must be updated on all write operations, overall disk performance may be reduced.

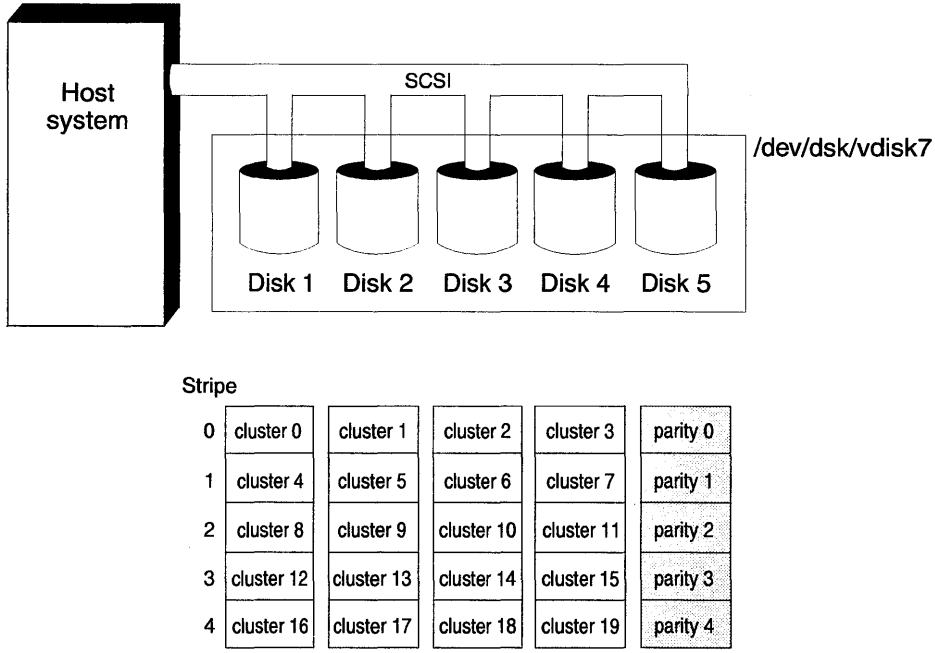


Figure 8-3 Example striped disk (RAID level 4)

Figure 8-3 shows an example of a RAID 4 configuration with Disk 5 as the parity disk and a cluster size of 32KB. (Note that disk blocks are 512 bytes long, so a 32KB cluster size corresponds to 64 blocks.)

Here is an example of an entry in */etc/dktab* file which shows the definition for the RAID 4 configuration tuned for group reads and writes.

```
/dev/dsk/vdisk7  array  5  64
    /dev/dsk/1s1   2000  97840
    /dev/dsk/2s1   2000  97840
    /dev/dsk/3s1   2000  97840
    /dev/dsk/4s1   2000  97840
    /dev/dsk/5s1   2000  97840 parity
```

Block-interleaved distributed parity array (RAID 5)

RAID 5 is based on data striping (as in RAID 0) and parity (as in RAID 4). The difference between RAID level 5 and RAID level 4 is that parity is striped across all disks. Because parity information is distributed, no one disk bears an excessive I/O load. For this reason, RAID 5 is preferred to RAID 4 for most I/O intensive applications.

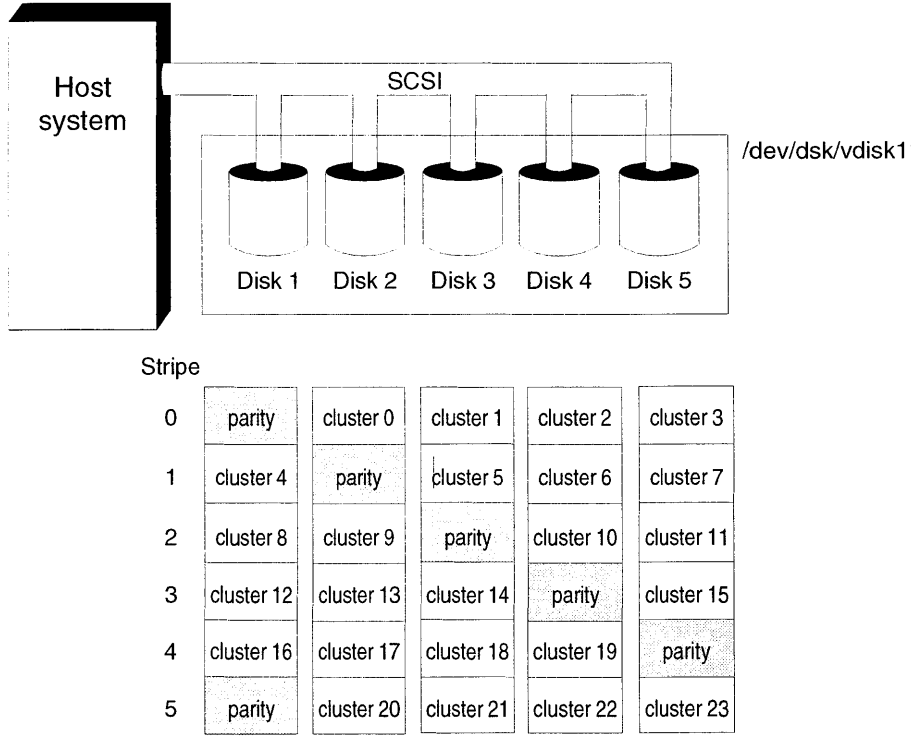


Figure 8-4 Example data and parity striped disk (RAID 5)

Here is an example of an entry for the RAID 5 configuration:

```
/dev/dsk/vdisk11  array  5    16
    /dev/dsk/0s1  2000  6864
    /dev/dsk/1s1  2000  6864
    /dev/dsk/2s1  2000  6864
    /dev/dsk/3s1  2000  6864
    /dev/dsk/4s1  2000  6864
```

Striped, mirrored array (RAID 10)

Striped mirrors (or RAID 10) can be built by striping mirrored disks, which are usually duplexed. Duplexing is achieved by setting up the disks on two independent buses.

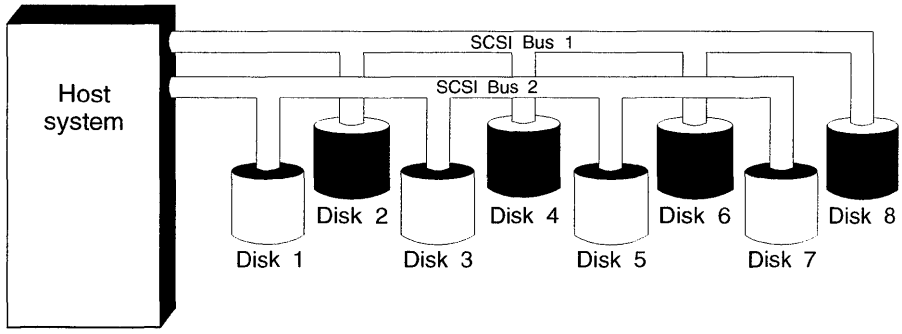


Figure 8-5 Example striped mirrored disk

In this example, Disk 1 is mirrored to Disk 2, Disk 3 is mirrored to Disk 4, and so on. The four mirrored virtual disks are then configured as a single striped disk. Disks 1, 3, 5, and 7 are striped together and disks 2, 4, 6, and 8 mirror them under a single virtual disk node. This configuration will protect disks from:

- Failure of one of the host adapters.
- Multiple disk failures on a single bus.
- Multiple disk failures on either bus.

This configuration will not protect from simultaneous disk failures of Disk 1 and Disk 2 or Disk 3 and Disk 4, and so on. Striped-mirror configurations can result in performance improvements over standard mirror or standard striped configurations.

Striped array of arrays (RAID 53)

Striped-arrays (or RAID 53) can be built by striping arrays. An example is shown in Figure 8-6, "Example striped disk arrays (RAID 53)" (this page).

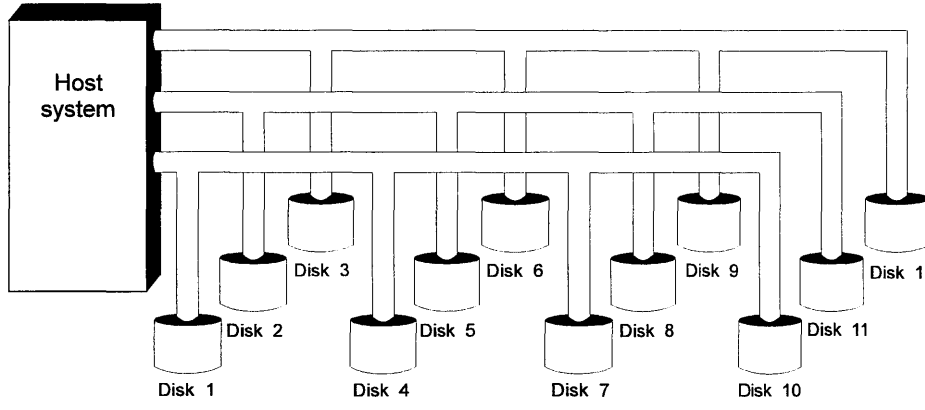


Figure 8-6 Example striped disk arrays (RAID 53)

In this example, Disk 1, Disk 2 and Disk 3 are configured as a RAID 5 array, Disk 4, Disk 5 and Disk 6 make up the next array, and so on. The four virtual disks are then configured as one striped disk.

This configuration will protect your disks from:

- Failure of one of the host adapters.
- Multiple disk failures on a single bus.
- Multiple disk failures on any bus (unless both disks are in a RAID 5 array).

How configuration information is stored

The configuration of a virtual disk system is specified in the file */etc/dktab* on the *root* partition (see *dktab*(F) for further details). This file is critically important; without it, there is no way of knowing which disk pieces are allocated to which virtual disks.

The *root* partition cannot be moved to a disk array, but to reduce the risk of losing the configuration file, it is possible to mirror the configuration file on a separate configuration database disk piece. (Additional space on the disk is given over to disk pieces that can be used by the system for other purposes, such as filesystem migration.)

The configuration database disk is similar to a concatenated virtual disk. The pieces in the virtual disk should be assigned as *dktab* or *spool*. Online reconfiguration is supported for the configuration database; pieces can be removed, added, or redefined. Figure 8-7, "Example configuration database" (this page) shows a configuration database with 1 *dktab* piece and 2 *spool* pieces. The database virtual disk type is reserved for virtual disk *vdisk0*.

NOTE *root* and *swap* divisions can only be configured as mirror virtual disks.

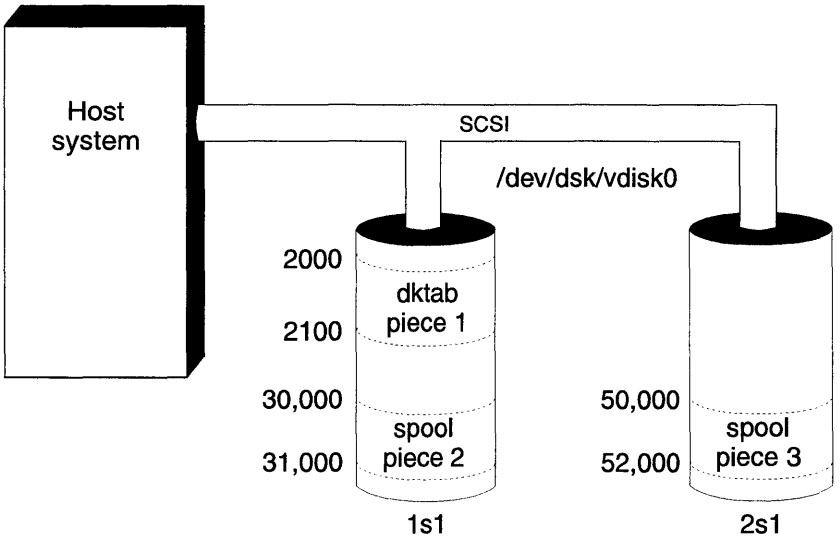


Figure 8-7 Example configuration database

The *dktab* piece on *vdisk0* is a backup copy of the */etc/dktab* file. It contains the current configuration of all the virtual disks on the system. Whenever the virtual disks are reconfigured, the backup copy of */etc/dktab* is updated. This provides a backup of the configuration information for the system. If the */etc/dktab* file is corrupted, it will automatically be restored from the configuration backup when the system reboots.

The configuration information stored in the database can also be manually retrieved at any time by selecting **Examine Contents** from the **Database** menu in the **Virtual Disk Manager**. The most recent configuration information which was saved will be displayed in the same format as */etc/dktab*.

To stop the system from using the configuration information in the *dktab* piece on system reboot, use **dkconfig -U**. See the **dkconfig**(ADM) manual page for more information.

The *spool* piece(s) permit fail-safe online reconfiguration. Whenever you reconfigure a virtual disk dynamically, a *spool* piece from *vdisk0* (if available) is used to provide space for the data. Having the *spool* piece available for online reconfiguration provides a mechanism to restart the reconfiguration in the event of an interruption, such as a system shutdown or panic.

When a fail-safe reconfiguration is interrupted, the next attempt to configure that virtual disk (done automatically at boot time) will restart the reconfiguration process in a fail-safe mode, from the point where it was interrupted. Ensure that at least as many *spool* pieces exist as the maximum number of simultaneous reconfigurations that might occur during normal operation (for example, if the system recovers from a panic and attempts to reconfigure all its virtual disks simultaneously). The *spool* pieces should also be at least three times the largest stripe width.

This is an example *vdisk0* definition with 3 *spool* pieces and 1 *dktab* piece:

```
/dev/dsk/vdisk0          config          4
        /dev/dsk/1s1     2000    2100    dktab
        /dev/dsk/3s1     2000    3000    spool
        /dev/dsk/4s3     2000    3000    spool
        /dev/dsk/4s3     3000    4000    spool
```

Be aware that *spool* pieces can be defined on the same physical disk partition and can be different lengths if space is available. The recommended length for the *dktab* piece is a minimum of 50 sectors, and 75 sectors for the *spool* pieces.

If there are no *spool* pieces defined in *vdisk0*, the reconfiguration operation will still be executed, but cannot be restarted if interrupted.

When using *spool* pieces to provide automatic recovery, reconfiguration performance will be slower.

Planning your system layout with virtual disks

Careful planning may help avoid errors which could lead to data loss. You may find the following sections useful in helping to plan your requirements.

- "Application and filesystem requirements" (this page)
- "Performance and reliability requirements" (page 372)

Application and filesystem requirements

Before you select the virtual disk type for your system, you must first know about your applications and how they interact with files.

Because a system can use multiple virtual disk types, you need to consider the I/O patterns across the entire storage space when defining your system layout. Use the **sar**(ADM) utility to get cumulative numbers for reads and writes for each physical disk.

Performance improvements will depend on a number of different variables.

Ratio of reads to writes

RAID 5 and RAID 4 virtual disks are particularly sensitive to the ratio of reads to writes. When writing data, they have to generate parity information; therefore they are better in read-intensive situations. At 100% reads, the performance of RAID 5/RAID 4 is high. Because the percentage of writes increases to 50%, the performance of RAID 5/RAID 4 can approach the performance of mirrored virtual disks.

RAID 5 provides the best read-only performance of the available virtual disk types. Generally, as the number of disks in an array increases, performance will increase (as more blocks are read/written in parallel.) Mirrored performance is better than a simple virtual disk because data is striped across both disks.

In read/write applications, striped virtual disks provide the best overall performance, but do not provide protection against disk failures. The performance of RAID 5 virtual disks suffers because of the overhead of parity generation when writing data; this does not occur during read-only access. Reduced performance on writing is a characteristic of RAID 5 arrays. As the percentage of write throughput increases, the relative performance of RAID 5 virtual disks decreases. As writes approach 100%, performance may fall below that of simple virtual disks. Mirroring may be a better performance alternative for write I/O rates >70%.

If performance is the highest priority, mirrored or RAID 10 virtual disks provide better performance than RAID 5/RAID 4 virtual disks as the percentage of write I/O increases (>70%).

For high I/O rates, striped virtual disks will provide the best performance compared to other virtual disk types, but without protection from data loss.

Dominant I/O size

This refers to the typical size of reads or writes made by applications. The combination of I/O size and cluster size can affect performance for RAID 0 (striped), RAID 5, and RAID 4 virtual disks. Choose a cluster size that optimizes performance; ideally it should be just larger than the dominant I/O size, so that an entire I/O transaction will fit in one cluster.

Distribution of I/O

Virtual disks can be used to obtain more efficient use of storage by balancing I/O evenly across all storage. You can monitor the balance of I/O at three different levels:

- Within the context of a single physical disk.
- Within the context of a single virtual disk (especially RAID 0, RAID 5, and RAID 4).
- Across all storage in the system (use **sar -d** to measure this).

Unbalanced I/O means that a lot of I/O activity is taking place on just one or two disks, causing a performance bottleneck. You can use **sar**(ADM) or select **Examine Performance** from the **Disk** menu of the **Virtual Disk Manager** to monitor I/O patterns.

Attempt to balance I/O evenly at each of these three levels. When unbalanced I/O patterns are identified, you can select **Modify** from the **Disk** menu of the **Virtual Disk Manager** to move data and reduce the imbalance.

Number of disks in a system configuration

Increasing the number of disks in a system configuration makes balancing I/O between disks easier. Many small disks provide better performance (and more flexibility) than a few larger disks.

Larger stripe widths (number of disks in parallel) improve the performance of RAID 0 (striped), RAID 5, and RAID 4 virtual disks.

Disk performance

This depends on a number of factors including disk platter rotation rate, disk head seek time, quality of the controller, and the size of the disk buffer. As much as possible, configure virtual disks with disks that have similar performance characteristics. Disks with different performance characteristics can be configured together, but the performance of the virtual disk will be limited by the slowest disk in the configuration.

Number of host controllers

Generally, the more host controllers in the system, the better. Try to distribute disks evenly across host adapters in configurations where data striping will be used extensively.

Host adapter caching

In general, the less random the I/O, the greater the benefits of caching. If random access data availability is more important than the performance benefits of host adapter caching, you may not need a caching host adapter.

Amount of system memory

The virtual disk driver uses system memory to manage the configured virtual disks. The more RAID 1, 4, and 5 virtual disks are configured, the more memory the system requires. On a lightly loaded system with few virtual disks, approximately 500KB is needed; on a fully configured, busy server an extra megabyte of memory is required.

Stripe width of a virtual disk

Increasing the stripe width (number of disks) of an array generally increases I/O performance.

Cluster size

Cluster size is the parameter that has the most impact on performance of array systems (RAID 0, RAID 4, or RAID 5). Improperly matched I/O and cluster sizes can adversely affect performance. When selecting a cluster size, the **Virtual Disk Manager** offers you some typical figures. These are approximations based on tested results, but may not be ideal for all applications. Performance may vary due to pattern of I/O and read versus write percentage.

Performance and reliability requirements

As systems with faster CPUs and increasing memory become available, disk I/O is rapidly becoming the primary bottleneck on overall system performance. Disk operations typically take milliseconds to complete; memory and CPU operations take nanoseconds to microseconds. Thus, a single disk block read may take as long as many thousands of CPU operations.

By making multiple disks into disk arrays, disk I/O speed is significantly improved because the time spent physically reading or writing the disk is reduced. For example, on network servers with many users accessing many small files, disk arrays allow multiple, independent I/O requests to be serviced in parallel by separate disks. The more disks in the array, the better the potential performance benefits.

Planning for increased reliability

Use redundancy and fault tolerance to increase reliability. Reliability is defined in this context as the ability to directly access data, even during drive failures. Redundancy provides data reliability and fault tolerance provides disk reliability.

Mirroring (RAID 1) is more expensive than other techniques, but may provide better performance than RAID levels 4 or 5 in write-intensive situations. For example, mirroring twenty 1GB drives requires forty 1GB drives; only twenty-one 1GB drives are required for a "20 + 1" RAID 5 array to provide an equivalent level of data reliability, but the RAID 1 configuration offers greater throughput.

The Virtual Disk Manager interface

The **Virtual Disk Manager** allows you to administer the virtual disks on your system. You can start the **Virtual Disk Manager** in any of the these ways:

- Double-click on the **Virtual Disk Manager** icon in the System Administration window on the Desktop.
- Start SCOadmin by entering **scoadmin** on the command line; select **Filesystems**, then select **Virtual Disks**.
- Enter **scoadmin Virtual Disk Manager** on the command line, (or abbreviate to **scoadmin virt**).

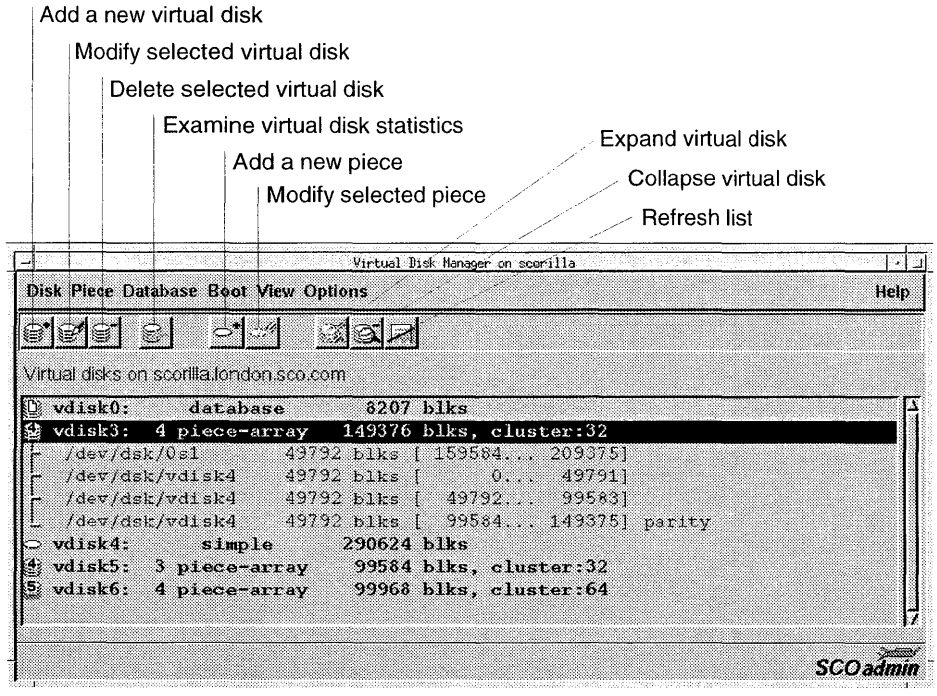


Figure 8-8 Main Virtual Disk Manager window

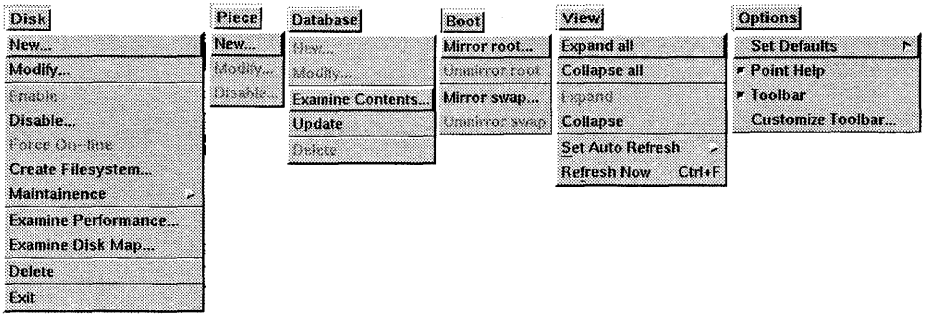


Figure 8-9 Virtual Disk Manager menus

NOTE Not all systems are licensed to use the **Virtual Disk Manager**. If your system is not licensed, the **Virtual Disk Manager** will exit with an error message whenever you attempt to run it.

For more information on using SCOadmin managers, see "Administering your system with SCOadmin" in the *SCO OpenServer Handbook*.

Adding virtual disks

To add a virtual disk, you create a virtual disk, allocate pieces of physical disks to the virtual disk, then mount the virtual disk.

To create a virtual disk, run the **Virtual Disk Manager**, and select **New** from the **Disk** menu. Accept the default values provided for the **Vdisk type** and **Device** entries or change the selections to suit your system requirements. Once you have created a virtual disk filesystem, you may mount it using the **Filesystem Manager**, as described in "Mounting and unmounting filesystems" (page 000).

NOTE Because virtual disks are intended for systems with more than one hard disk, your secondary disks must be configured as described in "Adding secondary hard disks" in the *SCO OpenServer Handbook*. When configuring these secondary disks with **mkdev**(ADM), do not run **divvy**(ADM); the **Virtual Disk Manager** does not use the disk divisions created by this utility except when migrating existing divisions to virtual disks.

To add a virtual disk:

1. Allocate one or more disk pieces for the virtual disk:
   - Select a disk piece and click on **Allocate piece**. Select a device (physical disk) for the piece. Using the disk map (showing the virtual disk layout), select a free area of the disk and assign it to the piece.

Disk pieces are located by their offset (in blocks, from block 0 of the partition) and their size. So, to create a 1000-block disk piece you might select the first free section of */dev/dsk/1s1* (the first partition on the second physical hard disk) and enter the value "1000" into the "Length" field.

- Click on **OK** to confirm your piece definition.

2. Continue allocating pieces until all the pieces in the virtual disk are defined.
3. Click on **Create** to create the virtual disk. If the virtual disk type uses parity data (RAID 1, 4, and 5) you will be prompted whether you want to restore parity at this time. You are prompted for the type of filesystem to place on the virtual disk. You have the option of creating a filesystem at this time, or later; see "Creating filesystems on virtual disks" (page 386) for details.

The **Virtual Disk Manager** permits you to create any configuration of virtual disk, and to allocate pieces to the disks as you like. Not all possible virtual disks make sense. For example, you can create a striped disk and specify that all the pieces reside on one physical hard disk; this will, however, lead to extremely poor performance (as the virtual disk will be unable to read or write data in parallel across the stripe).

See also:

- "Allocating or modifying disk pieces" (this page)

Allocating or modifying disk pieces

You can allocate disk pieces either while creating a new virtual disk, or by selecting an existing virtual disk (of a type that supports more than one piece) and choosing **New** from the **Piece** menu in the **Virtual Disk Manager**.

You can specify these parameters:

- The device the new piece resides on (physical partition or existing virtual disk).

  A list of available physical disk devices is displayed. You must choose a disk to place the piece on. When you select a disk, a map is displayed.
- The offset value within the selected device in 512-byte blocks.
- The length of the piece in 512-byte blocks.

The disk map shows which parts of the device have already been assigned to a virtual disk and which are free. It shows the virtual disk pieces allocated on the device, and the disk divisions allocated on the device. (Note that space allocated to **divvy** divisions can only be assigned to a simple virtual disk.)

NOTE You cannot allocate the free section of the root disk (*/dev/dsk/0s1*) shown on the disk map; this is used by the *root* filesystem. You cannot allocate free space that has been given over to divisions, created using **divvy**(ADM), as the division mechanism is incompatible with virtual disks; see "Converting filesystems to virtual disks" (page 387).

You can select a free section in the disk map; the offset and length fields will be updated appropriately. (You can also manually reduce the length field from the maximum size available, which is selected by default.)

If you are creating a simple virtual disk, you can overlay a single division with a single disk piece, in order to convert the division into a virtual disk. (Once a division is overlaid by a simple virtual disk, the division is deleted.)

For RAID 0, 1, 4 and 5 arrays, each piece must be of identical size. The first time you create a piece for such a disk, the piece length is fixed; all subsequent pieces you add to the array are forced to that size.

The **Virtual Disk Manager** software does not allow disk pieces to be added to an existing RAID 10 or RAID 53 configuration. If you need to add another piece, create the extra mirror or array piece with the **Virtual Disk Manager**, then exit and edit */etc/dktab* manually to add the extra piece. The format of this file is described in *dktab*(F). You will need to move the definition of the new virtual disk piece so that it occurs before the definition of the virtual disk that uses it.

If the virtual disk has a cluster size defined, then the disk length (size) must be a multiple of the cluster size. If this is not the case then the length is decreased appropriately and you are prompted to confirm this new size.

All pieces must be allocated before the virtual disk can be created.

NOTE When you add a new piece to an array that has out-of-date parity or is out-of-service, a parity restore will be started automatically after the reconfiguration has completed. This is not shown by the progress slider bar. The list of virtual disks is refreshed when the progress slider bar reaches 100% and the reconfigured virtual disk will be marked as having bad parity. When the restore has completed and the display refresh interval has elapsed, the reconfigured virtual disk will be shown with up-to-date parity (or no pieces out-of-service).

Creating nested virtual disks

You can nest virtual disks inside one another provided that you do not nest a virtual disk that has redundancy (mirrors and RAID 4/5) inside another virtual disk with redundancy. Doing this may cause the virtual disk driver to hang and suspend access to all configured virtual disks.

Three-way mirrors (created by nesting a mirror within a mirror) are an exception. You can configure these provided the master piece of the top level mirror is the second mirror. To configure *vdisk5* as a 3-way mirror in the */etc/dktab* file:

```
/dev/dsk/vdisk4 mirror  2       32
        /dev/dsk/1s1    2016    3200
        /dev/dsk/2s1    2016    3200

/dev/dsk/vdisk5 mirror  2       32
        /dev/dsk/vdisk4 0       3200
        /dev/dsk/3s1    2016    3200
```

The **Virtual Disk Manager** nests the disk definitions in */etc/dktab* correctly. If you edit */etc/dktab* manually, ensure that definitions of virtual disks used as pieces occur before the definition of the virtual disk that references them.

Adding a configuration backup

A configuration backup virtual disk contains a database of backup information to help the system recover following a disk error. The configuration backup virtual disk is also referred to as a configuration database virtual disk.

In the **Virtual Disk Manager**, create a configuration backup:

1. Select **New** from the **Database** menu.
2. Select **Yes** to include a *dktab* piece. If required, add a number of *spool* (page 367) pieces at this point and click on **OK** to continue to allocate disk space for the disk pieces.
3. Allocate pieces for the database, as for any other virtual disk, and create the configuration backup.

The configuration backup is visible in the **Virtual Disk Manager** as a virtual disk of the type **database**. To examine the contents of the database, select **Examine Contents** from the **Database** menu.

To modify a configuration backup, select **Modify** from the **Database** menu in the **Virtual Disk Manager**. (The **New** item is not available if a database exists on the system).

See also:

- "How configuration information is stored" (page 367).

Mirroring boot, swap, and root onto virtual disks

You may want to mirror your system's boot, swap, and root divisions onto a virtual disk, to provide backup for them. You cannot install the operating system itself onto a disk array, but mirroring provides protection against intermittent disk problems. There are two possible configurations:

Mirroring the boot, swap, and root divisions on a secondary hard disk

This configuration protects against disk I/O errors and it also allows you to reboot your system from the parity disk should the root disk fail.

If you use SCSI disks, the extended SCSI BIOS on most systems can detect the first two disks in the system, and you can sometimes configure this to boot the system from either disk. We recommend that you set up the parity disk as the second SCSI disk in the system. It may also be possible to attach the data and parity disks to separate SCSI buses provided that the host adapters are of the same type.

The storage capacity of the secondary disk must be at least as large as the root disk because the divisions must have the same start and end blocks.

NOTE We recommend that the root and parity disks have identical capacity, performance, and type (IDE, EIDE, or SCSI). Ideally, use the same model from the same manufacturer for each disk.

The parity disk must have the same size root partition as the root disk. The root partition must not contain any divisions or valid timestamps.

1. If the parity disk does not contain a valid root partition, use the **fdisk**(ADM) command to define the partition. For example, enter **fdisk -f /dev/rdsk/0s0** to discover the size of the root partition on the root disk. If the replacement disk is the second hard disk defined in the system, enter **fdisk -f /dev/rdsk/1s0** to define a root partition on it.
2. In the **Virtual Disk Manager**, select **Mirror** from the **Boot** menu.
3. In the **Select Divisions** window, select **Yes** for each of the root, swap, and boot divisions. Click on **OK** to accept this.

4. In the **Piece Allocation** window, select **Yes** to choose automatic piece allocation. Click on **OK** to accept this.
5. In the **Vdisk parameters** window, click on **OK** to accept the default cluster size of 32. Change the value if you want to use a different cluster size for the mirrored divisions.
6. Click on **Allocate pieces** to view the parity piece that will be used to mirror the three divisions. Click on **OK** to accept this.
7. Click on **Create** to make the parity piece, division tables, and masterboot block on the parity disk.
8. After the kernel has been relinked, exit from the **Virtual Disk Manager**.
9. Shut down and reboot the system.

Mirroring any combination of the boot, swap, and root divisions individually onto parity disk pieces

The pieces do not have to exist on the same physical disk as each other, and they can be defined in any order on the same physical disk. Mirroring the divisions in this way only protects against disk I/O errors. You cannot boot your system from these disk pieces. If your root disk fails, you must reinstall the operating system and restore the boot and root filesystems from a backup.

1. In the **Virtual Disk Manager**, select **Mirror** from the **Boot** menu.
2. In the **Select Divisions** window, select which of the root, swap, and boot divisions you want to mirror. Click on **OK** to accept this. If you choose to mirror all three divisions, select **No** in the **Piece Allocation** window to reject automatic piece allocation.
3. Define the individual disk pieces that will be used to mirror each division. See "Allocating or modifying disk pieces" (page 375).
4. Allocate a disk piece to each division, and click on **Create** to write the configuration changes to disk.
5. After the kernel has been relinked and you have defined parity disks for all the divisions, exit from the **Virtual Disk Manager**.
6. Shut down and reboot the system.

NOTE Although the boot and root parity disks are updated when the system is rebooted, the swap parity disk is not updated because it does not contain any useful information.

Booting your system from a parity disk

If you choose to create a mirror of the root disk, it is possible to reboot a system from the parity disk if the root disk fails. To do this, you may have to reconfigure the system's hardware so that it recognizes the parity disk as the root disk:

IDE or EIDE disks
: Remove the failed disk drive and configure the disk containing the mirror of the root disk as the "Master" device on the "Primary" controller.

SCSI disks
: Remove the failed disk drive and configure the disk containing the mirror of the root disk with the same SCSI ID number as the root disk. Note that on most systems, the SCSI ID of the root disk is 0; on some IBM systems, the SCSI ID of the root disk is 6.

  Alternatively, the extended SCSI BIOS on most systems can detect both root and parity disks when the machine is first turned on or reset, provided that both disks are attached to the same SCSI bus. If the root disk has failed and has been physically removed, it is usually possible to boot from the parity disk without reconfiguring its SCSI ID.

  If the root disk has failed and has not been removed, and the BIOS can still detect its presence, the system will not be able to boot if the masterboot program on the root disk has become corrupted. In this case, use an emergency boot floppy to boot the system and specify an appropriate bootstring at the `Boot` prompt.

  The following example bootstring defines that the executable kernel is located on the second hard disk defined in the system, that this disk is controlled by the SCSI disk driver, and that the root and swap divisions are accessible using the **vdisk** driver:

```
hd(72)unix hd=Sdsk root=vdisk(1) swap=vdisk(2) dump=none
```

  For more information, see **bootstring**(HW).

Unmirroring boot, swap, or root

To remove a mirror from one or more of the root, swap, or boot divisions:

1. In the **Virtual Disk Manager**, select **Unmirror** from the **Boot** menu.
2. In the **Select Mirrors** window, select which of the root, swap, and boot divisions you want to remove mirroring from. Click on **OK** to accept this.
3. After the kernel has been relinked, exit from the **Virtual Disk Manager**.
4. Shut down and reboot the system.

Replacing a mirrored root or parity disk

To replace a defective root or parity disk when a mirror disk is available:

1. Shut down the system and remove the failed disk drive.
2. If the root disk has failed, reconfigure the parity disk to have the same IDE or SCSI configuration as the root disk. Alternatively, for SCSI disks, you can use a boot floppy to reboot the system in steps 4 and 5. In this case, you must specify an appropriate bootstring at the `Boot` prompt to define where to find the bootable kernel, for example:

   ```
   hd(72)unix hd=Sdsk root=vdisk(1) swap=vdisk(2) dump=none
   ```

3. Install the replacement disk with the same IDE or SCSI configuration as the disk being replaced. If you reconfigured the parity disk as the root disk, configure the replacement disk as the new parity disk.

   NOTE The root partition on the replacement disk must be the same size and have the same partition number as the root partition on the original disk. The root partition must not contain any divisions or valid timestamps. If the disk has been used on another system, clear its partition, division, and timestamp tables before using it as a replacement disk.

4. If the replacement disk does not contain a valid root partition, reboot the system, place it in system maintenance mode, and use the **fdisk**(ADM) command to define the partition.
5. Reboot the system and place it in multiuser mode. The system should recognize the replaced disk and start to recreate the data on the mirror.
6. In the **Virtual Disk Manager**, select **Recreate** from the **Boot** menu.
7. In the **Select Mirrors** window, select the divisions whose division table entries you want to recreate. You will normally want to select **Yes** for all three divisions. Click on **OK** to make a new division table and masterboot block on the parity disk. This will allow you to boot from this disk if the root disk fails.

See also:

- "Planning your system layout with virtual disks" (page 369)
- "Adding virtual disks" (page 374)

Adding hot spares to virtual disks

The **Virtual Disk Manager** software supports the use of hot spare disks which are defined as pieces of an array. If one of the disks in the array fails, the spare will be brought online so that it can be reconstructed from the data and parity on the working disks.

In the **Virtual Disk Manager**, select the virtual disk. Select **Modify** from the **Disk** menu, and click on **Yes** in the **Configure a hot spare** item of the disk parameters.

You can also add hot spares to new disks when they are first configured.

Setting virtual disk defaults

You can adjust the default settings for virtual disks. This may be useful to you if you are configuring a large number of disks in a standard way.

In the **Virtual Disk Manager**, select **Set Defaults** from the **Options** menu, then click on **Vdisk Parameters** to set the default values for the different virtual disk types.

Table 8-2 Reference guide for setting disk parameters

Type	Pieces	Cluster	Hot spare
Simple	(1)		
Concatenate	(2)*		
Stripe(RAID 0)	(2)*	*	
Mirror(RAID 1)	(2)	*	*
RAID 4	(3)*	*	*
RAID 5	(3)*	*	*
RAID 10	(4)*	*	
RAID 53	(6)*	*	

The numbers in parentheses in the "Pieces" column indicates the minimum number of pieces which may be defined per type. The asterisks indicate whether a given option is available.

See also:

- "Virtual disk types" (page 359)
- "Allocating or modifying disk pieces" (page 375)
- "Clusters" (page 358)

Creating additional virtual disk nodes

The system creates 16 virtual disk nodes by default. If needed, additional virtual disk nodes can be created with **idmknod**. Edit */etc/conf/node.d/vdisk* to define an additional node, then use the command **/etc/conf/bin/idmknod** to recreate the nodes.

By convention, all virtual disk nodes have the form */dev/rdsk/vdisk** for character virtual disks and */dev/dsk/vdisk** for block virtual disks. The characters at the end of the name must be integers. For example, */dev/rdsk/vdisk17* is an example of a character virtual disk device and */dev/dsk/vdisk22* is an example of a block virtual disk device.

Do not use the following virtual disk device names, which are reserved for the purpose specified:

Name	Reserved for:
vdisk0	administration
vdisk1	mirroring root filesystem
vdisk2	mirroring the swap device
vdisk3	mirroring the boot device

Creating a RAID 10 virtual disk array

The following procedure creates a RAID 10 (striped, mirrored array) virtual disk configuration using previously configured mirrored virtual disks for its pieces. See "Striped, mirrored array (RAID 10)" (page 366) for more information about this type of array.

1. Configure two or more mirrored virtual disks using the **Virtual Disk Manager**.
2. Restore parity on each mirror.

   WARNING Do not create any filesystems at this stage.

3. Select **New** from the **Disk** menu, and choose to add a RAID 10 array.
4. Select the mirrored virtual disks that you created in step 1 for the pieces of the RAID 10 array.
5. The **Virtual Disk Manager** prompts you to create a filesystem on the new array.
6. Exit from the **Virtual Disk Manager**.

7. Use the **Filesystem Manager** to configure the filesystem's mount point and mount options.

Your RAID 10 array is now online and ready for use.

Creating a RAID 53 virtual disk array

The following procedure creates a RAID 53 (striped array of arrays) virtual disk configuration using previously configured RAID 5 virtual disk arrays for its pieces. See "Striped array of arrays (RAID 53)" (page 367) for more information about this type of array.

1. Configure two or more RAID 5 virtual disk arrays using the **Virtual Disk Manager**.
2. Restore parity on each RAID 5 array.

   WARNING Do not create any filesystems at this stage.

3. Select **New** from the **Disk** menu, and choose to add a RAID 53 array.
4. Select the RAID 5 arrays that you created in step 1 for the pieces of the RAID 53 array.
5. The **Virtual Disk Manager** prompts you to create a filesystem on the new array.
6. Exit from the **Virtual Disk Manager**.
7. Use the **Filesystem Manager** to configure the filesystem's mount point and mount options.

Your RAID 53 array is now online and ready for use.

Modifying virtual disks

To change virtual disk type, pieces, or size: in the **Virtual Disk Manager**, select **Modify** from the **Disk** menu, then select the appropriate item from the submenu.

If you want to migrate all your data to new pieces, the virtual disk type migrated to can be of any type (with the exception of RAID 10 and RAID 53). If all or some of the disk pieces in the new configuration are being used in the current configuration, certain migration paths are not supported.

Table 8-3 Guide for migrating to new types

New type	From Simple	From RAID 0	From RAID 4	From RAID 5	From mirror
Simple					*
Concatenate	*				
Stripe (RAID 0)	*		*	*	
Mirror (RAID 1)	*				
RAID 4	*	*		*	
RAID 5	*	*	*		

The asterisks in the table indicate where conversion from one type to another is possible. In these instances, new pieces are not being used.

NOTE Although it is possible to dynamically increase the size of a virtual disk, it is not possible to dynamically increase the size of a filesystem on the virtual disk. To increase the size of a filesystem, see "Creating filesystems on virtual disks" (page 386).

See also:

- "Planning your system layout with virtual disks" (page 369)
- "Adding virtual disks" (page 374)
- "Converting filesystems to virtual disks" (page 387)
- "Adding hot spares to virtual disks" (page 382)
- "Deleting virtual disks" (page 386)

Examining the current configuration

You can view the disk configuration in two ways: as a physical disk map (showing the allocation of physical disks to virtual disk pieces and divisions) or as an outline of virtual disk configuration (showing the allocation of virtual disk pieces to physical disks).

To view the physical disk map, select **Examine Disk Map** from the **Disk** menu of the **Virtual Disk Manager**. The physical disk map lists the partitions on each physical hard disk, along with the virtual disk pieces and divisions allocated from those partitions (and the virtual disks to which those pieces belong).

The virtual disk configuration is displayed in the **Virtual Disk Manager**. It provides an outline view of the virtual disks. You can expand or collapse the individual disks (using **Expand** or **Collapse** from the **View** menu) to see the pieces allocated to the disks. To expand the entire outline, select **Expand all** or **Collapse all** from the **View** menu. To expand the outline by default, select **Initial view expanded** from the **Set Defaults** item of the **Options** menu.

Deleting virtual disks

In the **Virtual Disk Manager**, select the virtual disk required, then select **Delete** from the **Disk** menu.

For information about securing your root and swap partitions, see "Mirroring boot, swap, and root onto virtual disks" (page 378).

For information about securing the configuration information that enables the **Virtual Disk Manager** to operate, see "Adding a configuration backup" (page 377).

Creating filesystems on virtual disks

In the **Virtual Disk Manager**, select a virtual disk, then select **Create Filesystem** from the **Disk** menu.

Select the filesystem type you require, then click on **OK** to create this new filesystem on the virtual disk. You are warned that existing data will be overwritten if the filesystem already exists on the virtual disk.

WARNING If you change the filesystem on a virtual disk, all data stored on the filesystem will be lost. Back up any data on the virtual disk before changing filesystems.

For example, to increase the size of a filesystem:

1. Back up the filesystem using the **Backup Manager**.
2. Unmount the filesystem using the **Filesystem Manager**.
3. Use the **Virtual Disk Manager** to increase the side of the virtual disk.
4. Create a new, larger filesystem (using the **Create Filesystem** item on the **Disk** menu).
5. Restore the filesystem data from the backup as described in "Restoring a scheduled filesystem backup" (page 129).

Converting filesystems to virtual disks

A traditional filesystem exists as a physical disk division. Because virtual disks provide more flexible storage than divisions, consider converting all your filesystem divisions to virtual disks. Once you have converted a division to a virtual disk, it will no longer be accessible via the divisions device nodes. It will be accessible via the virtual disk device nodes.

To migrate from a system based on divisions to virtual disks, unmount the filesystems on the division that you wish to convert, then create a simple virtual disk for each division using the **Virtual Disk Manager**, as described in "Adding virtual disks" (page 374). Allocate the division to the virtual disk, as described in "Allocating or modifying disk pieces" (page 375). The division will be deleted automatically once the virtual disk is assigned to it.

When the filesystem is in a simple virtual disk configuration, migration to other array configurations can be done without taking the disk offline. The division should not be accessed while it is migrated.

For your existing divisions:

1. Unmount the filesystems on the divisions that you want to convert.
2. For each division, select **New** from the **Disk** menu, add a simple one-piece virtual disk as described in "Adding virtual disks" (page 374), and allocate the division to the virtual disk. This overlays each division with a simple virtual disk.
3. For each division, select **Modify** from the **Disk** menu, and migrate the simple virtual disk to the desired RAID type. For example, if you want to create a mirrored filesystem, select **Mirror** and allocate a spare disk piece to the parity piece. If migrating to a disk array, you must have spare disk space for temporary storage.

4. Use the **Filesystem Manager** to update the information about mounted filesystems. Replace the filesystems' **divvy** device nodes with their virtual disk ones. For example, if you have replaced the */u* filesystem with a mirror *vdisk4*, mount */dev/dsk/vdisk4* instead of */dev/u* on the */u* mount point.

Tuning the performance of virtual disks

You can tune these parameters in the **Virtual Disk Manager**:

Cluster size: Cluster size affects the performance of a virtual disk. This effect is also dependent on the average size of I/O operations. Cluster sizes can be changed by reconfiguration, but the length must remain a multiple of the new cluster size. See "Distribution of I/O" (page 371).

Load balancing: If the I/O count is unbalanced, you can add additional disk pieces, move data or change the cluster size to balance the I/O across the array. Using the I/O statistics reported through the **Examine Performance** item of the **Disk** menu, you can track the level of activity of different drives used by a virtual disk.

See also:

- "Application and filesystem requirements" (page 369)
- "Tuning virtual disk performance" in the *Performance Guide*

Monitoring virtual disk performance

In the **Virtual Disk Manager**, select a virtual disk, select **Examine Performance** from the **Disk** menu, then click on **Refresh** to obtain current statistics.

A snapshot of disk I/O statistics and read/write ratios are displayed for the virtual disk and for individual disk pieces within it. "Application and filesystem requirements" (page 369) explains the statistics displayed.

Use the **Refresh** button to update the statistics or the **Zero** button to clear existing statistics.

Use this information to monitor I/O activity on your system. To reduce performance constraints, balance the I/O more evenly as described in "Application and filesystem requirements" (page 369). For more information on tuning your system, see "Tuning virtual disk performance" in the *Performance Guide*.

Troubleshooting virtual disks

You may find the following procedures and information useful when troubleshooting virtual disks:

- "Disabling and re-enabling virtual disks" (this page)
- "Forcing virtual disks online" (this page)
- "Checking and restoring parity data" (page 390)
- "Repairing a failed drive" (page 390)
- "Possible problems" (page 391)
- "Warning messages" (page 393)
- "Notice messages" (page 398)

Disabling and re-enabling virtual disks

In the **Virtual Disk Manager**, select **Enable** or **Disable** from the **Disk** menu.

The disable operation is supported on all virtual disks. You can use the disable operation to help replace or repair failed disk pieces.

Forcing virtual disks online

A virtual disk can go offline for varying reasons, as described in "Possible problems" (page 391); for example, when more than one piece is out of service (OOS) or when one piece is out of service and parity is out of date (OOD). Only RAID 1, RAID 4 and RAID 5 can go offline.

To force an offline disk back online:

1. In the **Virtual Disk Manager**, select a virtual disk which is offline.
2. Select **Disable** from the **Disk** menu.
3. Select **Force On-line** from the **Disk** menu.

If the virtual disk uses parity data, it will also need restoring. Because parity can only be used to recover data from a single disk failure, some data may be lost.

Checking and restoring parity data

You can check the disk for valid parity information, reconstruct parity (restore), or fix errors.

In the **Virtual Disk Manager**, select a virtual disk, then select **Maintenance** from the **Disk** menu.

Select either **Fast restore** or **Slow restore** from the **Maintenance** submenu of the **Disk** menu to update parity information when an array is initially configured, or to reconstruct data on a disk that has been repaired. The virtual disk remains accessible during this restore operation.

Select **Check Parity** from the **Maintenance** submenu to ensure that disk data and parity are correctly matched.

If a drive fails and the system is offline for repairs, see Chapter 17, "Adding hard disks" in the *SCO OpenServer Handbook* for more information on replacing or repairing it.

In event of a restart following an unscheduled shutdown, parity information is automatically restored when the system reaches run level 2 (goes multi-user). Parity restoration on reboot can be either slow or fast. A fast restore reduces the period during which the virtual disk is vulnerable to a second failure, but impacts system performance; a slow restore has little impact on system performance but takes significantly longer. Automatic parity restoration is controlled by */etc/rc2.d/S81VDRESTORE*; you can change the speed of restoration by editing this shell script.

Repairing a failed drive

A disk piece may be taken out of service for a number of reasons (controller or bus errors, bad cables or connectors, power failure, surface defects or other hardware problems). In the case of surface defects, run **badtrk**(ADM) to remap the bad blocks. Usually the drive that failed can be brought back into service by restoring the parity on the failed drive (this page). If the drive needs repairing and the system does not support hot insertion (replacement of a drive unit while the array is operating), you must shut down the system to replace the drive.

NOTE If the RAID configuration does not include a hot spare, use the **Virtual Disk Manager** to swap the failed drive with another drive to prevent the RAID virtual disk from going offline due to a second I/O error. Replacement drives must be fully configured with **mkdev hd** before they can be used in virtual disks.

When the system is booted, the array will be online and the disk piece associated with the failed drive will be out of service. If a standby disk was configured, the applications will be running on the spare. The applications will continue running without interruption as you start to configure the replacement drive.

Once the drive is configured, the RAID array can recreate the data on the replacement drive. If the configuration is running on a standby disk (spare) the restore will update the data on the replacement drive and the spare will be placed back on standby.

Possible problems

A virtual disk can be online or offline. In the online state, the virtual disk is active and all data is accessible.

If a failure occurs, appropriate console warning messages and status information will be displayed. If a single drive fails on a RAID array (levels 1, 4 and 5), the virtual disk will remain online and all data will be accessible; in other circumstances, the virtual disk may go offline. Simple, concatenated and stripe virtual disk types always remain online as disk errors are passed back to the application.

Status information is displayed in the **Virtual Disk Manager** after the disk or piece information. Possible error states are:

OUT-OF-SERVICE (OOS)
: identifies a disk failure. One of the pieces in the array is inaccessible. If a single piece is in the OOS state, the array remains online.

OUT-OF-DATE (OOD)
: identifies a piece with corrupt parity. The array will remain online, unless a disk failure occurs. A restore operation should be performed as soon as possible. If a disk fails when one of the disk pieces in the array is in the OOD state, the entire array will go offline.

spare
: indicates which disk piece is the hot spare (when configured). If the hot spare is in use, the status indicator IN USE is added. The hot spare piece will automatically replace the piece identified as OOS.

BAD PARITY
: indicates a virtual disk with bad parity. This might also mean a new disk which has not had a restore operation, or an array that has just been brought online.

The array enters the offline state when virtual disk I/O accesses fail. When an array goes offline, repair the virtual disk and restore data from a backup.

See also:

- "Invalid timestamp on root device mirror" (this page)
- "Mirror root failure" (this page)
- "Warning messages" (page 393)
- "Notice messages" (page 398)
- "Offline disk array" (this page)
- "Kernel virtual memory shortage" (page 393)

Invalid timestamp on root device mirror

The **Virtual Disk Manager** uses timestamps on RAID virtual disk configurations to ensure proper operation and data integrity. If a timestamp on one of the mirror virtual disk pieces becomes invalid, the piece will not be fully configured. You cannot set the timestamps to a known state on a mirror virtual disk that is online.

If this happens, the out-of-service piece on the mirror root device cannot be restored.

1. Unmirror the root device (page 378).
2. Shut down the system and reboot, as described in Chapter 10, "Starting and stopping the system" in the *SCO OpenServer Handbook*.
3. Mirror the root device again (page 378).

Mirror root failure

If the primary disk fails during the system reboot (when mirroring the *root* disk for the first time), the array will go offline and the system boot will fail. At this point in the boot sequence, the system cannot switch over to the secondary disk if it has not been completely restored. Before replacing the primary drive or rebuilding the system, remove power from the secondary disk and try to boot the system. If the primary disk is not completely bad, the system will boot. When the system boots, unmirror the root device (page 378). Once the problem has been corrected, try to mirror the root device again.

Offline disk array

When an array or mirrored virtual disk has a mounted filesystem and the array or mirror goes offline due to error conditions, the filesystem becomes unusable. At this point, the filesystem cannot be unmounted (much the same as a hard disk failure). The system must be rebooted to clear this condition.

An array or mirror may go offline when more than one piece is out of service or one piece is out of service and parity is out of date. To rectify this:

1. Disable the virtual disk (page 389).
2. Force the virtual disk online (page 389).
3. Restore the parity (page 390).
4. Restore the filesystem data from the backup as described in "Restoring a scheduled filesystem backup" (page 129).

If the system fails (crashes) while running on a RAID array, the parity data will be automatically regenerated when the system is booted. This will ensure that parity data accurately reflects the data on the other drives in the array. If an I/O error is encountered while the parity information is being restored, the array will go offline. It is recommended that a UPS (uninterruptable power supply) be used to reduce the risk of power outages on systems using virtual drives.

Kernel virtual memory shortage

When the system has many RAID virtual disk configurations, with large cluster sizes or a heavy I/O load, the performance of the array may be reduced due to the high contention for system resources (buffers, kernel virtual memory, and so on). By increasing the total amount of physical memory, the system and array performance can be improved. See "Warning messages" (this page) for more information on driver error messages related to kernel virtual memory.

Warning messages

Warning messages (error messages that begin with `WARNING:`) alert you to virtual disk failures that require immediate attention. These messages are recorded in */usr/adm/messages* and are displayed on the console.

vdisk *n*: too many levels of virtual disks

The virtual disk configuration is not supported. The driver failed the virtual disk because only eight levels of nesting are allowed by the disk array driver. The desired configuration exceeds that level.

vdisk *n*: failed to open piece *m*

There are two possibilities:

- The driver was unable to open piece *m* during reboot. If the virtual device is an array or mirror, the device is functional but the out-of-service disk drive must be replaced or repaired as soon as possible. After the drive is replaced, restore parity (page 390). If the virtual disk is not an array or mirror, the virtual disk is not functional. Replace or repair the failed disk drive and restore the data from backup.
- The initial configuration of the virtual disk failed. The driver was unable to open piece *m* of the virtual disk. If the virtual disk is an array or mirror, the device is functional. The disk piece is out of service or is not currently configured. The drive must be repaired or replaced or the disk piece must be usable to the system before enabling the virtual disk (page 389).

vdisk *n*: spare will not be operational

There are three possibilities:

- During configuration of an array or mirror, the driver failed to open the disk piece defined as the spare. The virtual disk is functional, but if a drive fails, the spare piece will not be brought online. If you wish to include the spare in this configuration:
  1. Disable the virtual disk (page 389).
  2. Repair or replace the spare drive.
  3. Force the virtual disk online (page 389).
  4. Restore the parity (page 390).
- The driver failed to access the spare piece when reading or writing the timestamp. The array or mirror is still functional, but the spare will not be brought online when a data piece fails. The spare drive must be repaired or replaced if you wish to include the spare in the virtual configuration.
- The array has bad parity or a piece is out-of-service when the array was enabled.

vdisk *n*: is being taken offline
vdisk *n*: piece *m* and piece *p* are out-of-service

The array or mirror is no longer functional. The driver was unable to access two disk pieces in the virtual disk or the driver was unable to access one disk piece in the virtual disk while the parity data was out of date. You must:

1. Disable the virtual disk (page 389).
2. Repair or replace the disabled drives.

3. Force the virtual disk online (page 389).
4. Restore the parity (page 390).
5. Restore the data for this virtual disk from backups as described in "Restoring a scheduled filesystem backup" (page 129).

vdisk: insufficient memory to ...

The system does not have enough memory to initialize the virtual disk driver or perform a driver operation. If this error message persists and there is no "memory leak" (a process that progressively uses up memory without releasing it), add additional physical memory to the system or reduce the virtual disk or other system resources. See "Virtual disks" in the *Performance Guide*.

vdisk *n*: failed to allocate kernel virtual memory

The system does not have enough memory to allocate the necessary buffers for I/O. If this error message persists, consider adding additional physical memory to the system or reducing system resources.

vdisk *n*: failed to read/write timestamp on piece *m*

The driver failed to access disk piece *m* when reading or writing the timestamp. Piece *m* will be taken out of service, but the array or mirror will still be functional.

Restore parity on the out-of-service piece (page 390). If the restore fails, the out-of-service drive should be repaired/replaced as soon as possible and the parity restored.

vdisk *n*: new configuration offline, repair out-of-service drive

The virtual disk in not functional. The driver failed to open or access a disk piece when initializing a new configuration. The disk drive is either disabled or the disk piece is not defined properly. Repair or replace the failed drive or ensure the disk piece is available, then force the virtual disk online (page 389). If the virtual disk is an array or mirror, restore the parity (page 390).

vdisk *n*: reconfiguration failed, restore from previous backup

There was a system crash while the reconfigure operation was in progress. The array is offline. You should:

1. Force the virtual disk online (page 389).
2. Restore the parity (page 390).
3. Restore the data for this virtual disk from backups as described in "Restoring a scheduled filesystem backup" (page 129).

vdisk *n*: too many pieces out of service [run dkconfig -cf]

The array or mirror is no longer functional. The driver was unable to access two disk pieces in the virtual disk. You should force the array or mirror online (page 389).

If the configuration fails, you should:

1. Repair or replace the two disabled drives.
2. Force the array or mirror online again.
3. Restore the parity (page 390).
4. Restore the data for this virtual disk from backups as described in "Restoring a scheduled filesystem backup" (page 129).

vdisk *n*: read invalid timestamp on piece *m*

The timestamp on disk piece *m* of the array or mirror has become desynchronized. You should:

1. Disable the virtual disk or mirror (page 389).
2. Force it online (page 389).
3. Restore the parity (page 390).

vdisk *n*: timestamps are not valid [run dkconfig -cf]

Timestamps for the disk pieces, which make up the array or mirror, are out of synchronization. One or more of the disk pieces, which make up the array or mirror, were accessed individually prior to reboot or an enable. You should:

1. Force the array or mirror online (page 389).
2. Restore the parity (page 390).

vdisk *n*: failed to bring spare online

The driver was unable to access a disk piece and attempted to replace the failed disk (the disk piece will be taken out of service) with the spare disk piece. While updating the data on the spare piece, an I/O error occurred. The spare piece will not replace the out-of-service piece in the virtual disk. The array or mirror is still functional, but you should replace the out-of-service data drive and spare drive as soon as possible.

vdisk *n*: timestamps not closed properly, parity must be restored

The virtual disk was fully operational prior to a system crash. Data on the parity piece may not be accurate. The system will automatically restore parity during boot, so no intervention is necessary in this case.

vdisk *n*: timestamps not closed properly, parity may be out-of-date

The virtual disk had one piece out of service prior to a system crash. Data on the virtual disk may not be accurate. Repair or replace the out-of-service drive and restore the virtual disk data from backups as described in "Restoring a scheduled filesystem backup" (page 129).

vdisk: failed to spawn vddaemon error = *x*

The daemon was not spawned during reboot. RAID virtual disks may not be operational. Reboot the system and use **ps -aef** to make sure the **vddaemon** is running. If the problem occurs again, the system may be corrupted. Deconfigure the virtual disks and reinstall the **Virtual Disk Manager** package.

At least one **vddaemon** must be running. On multiprocessor systems, one is started per CPU for extra performance. If only one is running, then virtual disk performance may be reduced.

vdisk *n*: piece *m* is out of service

The driver was unable to configure disk piece *m* of the virtual disk. Piece *m* will be taken out of service. The array or mirror is functional, but you should repair or replace the disk drive as soon as possible and restore the parity of the failed drive (page 390). See "Repairing a failed drive" (page 390) for more information.

vdisk *n*: piece *m* is being taken out of service

The driver failed to access disk piece *m* of the virtual disk. This disk piece will be taken out of service. The array or mirror is functional, but you should repair or replace the out-of-service disk drive as soon as possible and restore the parity (page 390) for the failed drive.

vdisk *n*: not enough jobs to restore from spare

The maximum number of outstanding jobs for the specified virtual device has been reached. Restore the parity again (page 390). See "Virtual disks" in the *Performance Guide*.

vdisk *n*: restart of reconfiguration failed, restore from previous backup

A reconfiguration on vdisk *n* was in progress and was interrupted. The driver attempted to restart the reconfiguration from the interrupted point, but could not. The virtual disk will be offline. You should:

1. Force the disk online (page 389).
2. Restore the data for this virtual disk from backups as described in "Restoring a scheduled filesystem backup" (page 129).

vdisk: write to data base piece *m* failed err = *x*

A write to *dktab* piece *m* of *vdisk0* failed. Configuration information is not valid on that backup piece. You can either:

- Modify the affected *dktab* piece and relocate it (page 385). The **Virtual Disk Manager** will then reconfigure *vdisk0*.
- Modify the database and remove the affected *dktab* piece (page 377).

vdisk *n*: cannot restart reconfiguration if interrupted

Reconfiguration of the indicated virtual disk cannot be restarted if it is interrupted before completion. Verify that enough *spool* pieces are configured in *vdisk0* for as many simultaneous fail-safe reconfigurations as will be executed. See "Adding a configuration backup" (page 377) for information on modifying the virtual disk database.

vdisk: piece *x* is too small to backup configuration information

The piece number *x* of *vdisk0* is too small to back up the current configuration of configured virtual disks. Increase the length of that piece or reconfigure *vdisk0* to include a larger disk piece. See "Adding a configuration backup" (page 377) for information on modifying the virtual disk database.

Notice messages

Notice messages (error messages that begin with `NOTICE:`) alert you to error conditions where recovery has already occurred, but some operator intervention is necessary. Notice messages are recorded in */usr/adm/messages* and are displayed on the console.

vdisk *n*: new configuration online, parity must be restored

A virtual disk has been configured for the first time.

vdisk *n*: bringing spare online

One of the data pieces failed in the virtual disk. The spare piece will be brought online to replace the failed drive.

vdisk *n*: spare is online

One of the data pieces failed in the virtual disk. The spare piece was brought online to replace the failed drive. Replace or repair the out-of-service data drive and restore the parity for the out-of-service disk piece (page 390). See "Repairing a failed drive" (page 390) for more information.

vdisk *n*: cannot bring spare online during reconfiguration

A disk piece was taken out of service in the current or new configuration during the reconfigure operation. Either the current or new configuration contains a spare. The spare piece will not be brought online during the reconfigure operation. After the reconfiguration completes, the spare will be operational.

vdisk *n*: corrected error on piece *m*, no data lost

A bad block was detected and the disk array driver was unable to access piece *m* of the virtual disk during its first attempt. The driver recreated the data, retried the job, and was able to access the disk piece successfully on the second attempt. The array or mirror is functional. Ignore all previous bad block messages displayed by the underlying disk driver.

vdisk: job time-out; restarting job processing

During error recovery, there was a job time-out for one of the disk pieces in the virtual disk. The disk array driver will take the disk piece out of service and continue processing all remaining jobs. Restore the parity (page 390) on the out-of-service disk piece.

vdisk: job pool is empty

The maximum number of outstanding jobs for the disk array driver has been reached. The driver will not process any more jobs until most of the outstanding jobs are complete. This limit is controlled by the **VDJOBS** kernel parameter. See "Using configure to change kernel resources" in the *Performance Guide* for information on changing kernel parameters.

vdisk *n*: job queue is full
vdisk *n*: piece pool is empty

The maximum number of outstanding jobs for the specified virtual device has been reached. The driver will not process any more jobs for this device until most of the outstanding jobs for the virtual disk are complete. This limit is controlled by the **VDUNITJOBS** kernel parameter under the **Virtual disks** item of performance tunables. See "Virtual disks" in the *Performance Guide*.

Appendix A

Customizing UNIX system startup

When your system is switched on and booted, certain aspects of the UNIX system operation are set up. The system reads initialization files at startup, when changing run levels, and whenever a user logs in. By modifying these files, you can adapt system startup.

The system initialization files contain commands or data that:

- set initial run levels
- set the system clock
- enable terminals
- start programs
- check and mount specified filesystems
- clean up temporary directories
- set home directories and terminal types for users
- display system messages

The files discussed here are */etc/inittab*, the scripts in the */etc/rc2.d* directory, the *.profile*, *.cshrc*, and *.login* files (along with their system-wide counterparts in */etc*), and the */etc/motd* file.

The system administrator can modify the startup files to create any initial system and user environment. For example, by adding or changing entries in the *inittab* file, specific terminals can be enabled (or disabled) when the system enters or leaves a particular run level. By changing a script in the */etc/rc2.d* directory, process accounting can be started automatically at system startup. The administrator can also customize a specific user's environment by modifying the *.profile* or *.login* file in their home directories.

The initialization files are ordinary text files and can be modified using a text editor. Entries in the */etc/inittab* file must follow a specific format described in the **inittab**(F) manual page. For more information on **init** run levels, refer to the **init**(M) manual page. The scripts in */etc/rc2.d* and the *.profile* and *.login* files contain Bourne shell commands and comments. See Chapter 11, "Automating frequent tasks" in the *Operating System User's Guide* for more information on shell programming.

Changing the /etc/inittab file

The **/etc/init** program starts during the last phase of kernel initialization and has a "process id" (PID) of "1". The **init** process starts all other processes. The */etc/inittab* file contains instructions for **init**. The **init** program reads the *inittab* file under three circumstances: at boot time, when an **init**-started process completes, and when the system administrator executes either the **/etc/init** or **/bin/telinit** command with a run-level argument. The arguments passed to **init** allow you to change the system run level or force **init** to examine the *inittab* file without changing the run level. When the system changes run levels, **init** scans *inittab* for instructions that apply to the new state.

NOTE When you modify *inittab*, the change is only temporary because each time the kernel is relinked, a new *inittab* file is created. To change the initialization procedure permanently, you must also modify the source from which the *inittab* file is recreated. To add a new entry, append it to the */etc/conf/cf.d/init.base* file. To modify an entry, locate and edit the existing entry in */etc/conf/cf.d/init.base* or in one of the other component files in the */etc/conf/init.d* directory.

The *inittab* file is made up of entries that contain four fields separated by colons:

label:*run_level*:*action*:*process*

Table A-1 inittab fields

Field	Description
label	a unique identification label of up to four characters
run_level	the **init** level at which the entry is executed
action	a keyword indicating the action that **init** is to take on the process
process	the process **init** executes upon entering the specified run level

If there is more than one run level specified for an *inittab* entry, the levels appear in the second field without separators. If the run level field is empty, the entry is executed in all numeric run levels (0-6). When the run level changes, any process that does not have an entry for the new run level

receives a warning signal (15). If the process does not terminate after 5 seconds, it receives a kill signal (9). The current state of the **init** process determines how it executes the *inittab* entry.

When the **init** program is initially invoked, it scans *inittab* for entries that contain the action keywords described in Table A-2 (this page). These entries are executed only during **init**'s boot-time read of *inittab*.

Table A-2 inittab single-user keywords

Keyword	Description
boot	starts the process and continues to the next entry without waiting for the process to complete. When the process dies, **init** does not restart the process.
bootwait	starts the process once and waits for it to terminate before going on to the next *inittab* entry.
initdefault	determines which **init** level to enter initially, using the highest number in the "run_level" field. If the "run_level" field is empty, **init** interprets the run level as **0123456** and enters run level **6**. If there is no **initdefault** entry in *inittab*, then **init** requests an initial run level from the user at boot time.
sysinit	starts the process the first time **init** reads the table and waits for it to terminate before going on to the next *inittab* entry. Entries with the **sysinit** keyword are executed before **init** tries to access the console.

When the run level changes from single-user to a numeric run level (0-6), **init** scans entries with the actions in Table A-3 (page 404) and executes only those entries with the appropriate "run_level" field set.

Table A-3 inittab run_level field keywords

Keyword	Description
off	sends a warning signal, waits 5 seconds, then sends the kill signal to the process if it is currently running. If the process is not running, it ignores the *inittab* entry.
once	starts the process and continues to the next entry without waiting for the process to complete. When the process dies, **init** does not restart the process.
ondemand	is functionally identical to **respawn**; the **ondemand** keyword is used only with the **a**, **b**, or **c** run-level values.
respawn	starts the process if it is not currently running and continues to the next entry without waiting for it to complete; restarts the process when it dies. If the process is already running, **init** ignores the *inittab* entry.
wait	starts the process and waits for it to complete before going on to the next *inittab* entry.

If the system hardware is capable of detecting power failure and **init** receives a power failure signal, **init** executes entries containing the actions shown in Table A-4 (this page) if the run level is appropriate.

Table A-4 inittab powerfail keywords

Keyword	Description
powerfail	starts the process once and continues to the next entry without waiting for the process to complete.
powerwait	starts the process once and waits for the process to complete before going on to the next entry in the table.

In the following example, the *inittab* entry sets the default run level for the system to single-user mode when **init** is initially invoked:

```
is:S:initdefault:
```

NOTE The **initdefault** action is not associated with any process. The third colon (:) in this entry is necessary; if it is missing, **init** ignores the entire entry.

You can configure your system to come up in multiuser mode by editing this *inittab* entry, changing the "S" in the run-level field to a "2".

The following *inittab* entry runs the **/etc/bcheckrc** script when the system is booted (or rebooted) and waits for the process to complete before processing the next entry. (The **bcheckrc** script checks the filesystems and sets the date.)

```
bchk::sysinit:/etc/bcheckrc </dev/console >/dev/console 2>&1
```

The **init** process also starts the **/etc/getty** program according to instructions in *inittab*. The **getty** ("get a tty") program sets up communication between the system and terminals. For more information, see the **getty**(M) manual page.

The following example *inittab* entry tells **init** to start a **getty** process (if one does not already exist) for *tty01* (the console) at 9600 baud when the current run level is 2:

```
co:2:respawn:/etc/getty tty01 sc_m
```

The **respawn** action instructs **init** to restart the getty process each time it dies and to continue processing the next *inittab* entry without waiting for the current process to complete. By changing the action from **respawn** to **wait**, you tell **init** to wait until the current process has finished before reading the next entry in *inittab*.

See the **inittab**(F) manual page for a detailed description of the format of the *inittab* file and an explanation of the action keywords.

To make your changes to *inittab* effective immediately, execute the **telinit Q** command. This command causes **init** to reexamine the modified *inittab* file without changing the run level.

Changing scripts in /etc/rc2.d

Upon entering **init** state 2 (multiuser mode) from either a higher **init** state (3-6) or from single-user mode, **init** executes the **/etc/rc2** script according to the instructions in */etc/inittab*. The **rc2** script sets certain environment variables and runs scripts in the */etc/rc2.d* directory. Some of the scripts in *rc2.d* run scripts in subdirectories of the *rc.d* directory.

This section describes the scripts in the */etc/rc2.d* directory that are run by **rc2** and explains the steps for adding your own script. The **rc2**(ADM) manual page describes the other scripts that **rc2** runs.

Table A-5 (page 406) gives a brief description of some of the scripts in */etc/rc2.d*.

Table A-5 /etc/rc2.d scripts

Script	Description
S00MDAC	starts Mylex disk array monitor
P00SYSINIT	starts kernel message logger
S00VDISK	configures virtual disk arrays
I01MOUNTFSYS	mounts filesystems specified in */etc/default/filesys*
P03RECOVERY	tidies up **vi** editing sessions after a crash
P04CLEAN	removes temporary files
P05RMTMPFILES	removes temporary files
P15HWDNLOAD	downloads hardware
P16KERNINIT	starts, process accounting, network, and other kernel initialization
P20sysetup	configures print system and generates */etc/systemid*
P21perf	starts system accounting
S25pm	starts power management event routing and servicing daemon
S35dlpi	configures the network card driver interface
P70uucp	cleans up UUCP lock files
P75cron	starts **cron** daemon
S80lp	starts **lpsched** and net utilities
S85tcp	starts TCP/IP, name service, and routing
P86mmdf	starts **mmdf** deliver daemon
P86scologin	starts **scologin**
P87USRDAEMON	starts user daemons
P88USRDEFINE	executes user-definable commands after boot
S89hostmib	starts host MIB service
S89nfs	starts NFS service
P90RESERVED	mails **fsck** output saved during autoboot to *root*
S93scohttpd	starts **SCOhelp** daemon
S95calserver	starts **calendar** daemon
S99apcssd	starts UPS port monitor

The */etc/rc2.d* directory on your system may contain scripts other than the ones listed in the table. This is because during installation, many add-on programs insert their own daemon-initialization scripts in this directory. This directory may also include scripts that clean up the temporary or lock files for an add-on program.

You can write your own scripts to run when the system enters **init** state 2. For example, you can write a script that sets up a RAM disk or starts a network and add it to */etc/rc2.d*.

The following factors should be considered when writing a system startup script to be placed in *rc2.d* :

- When going from a higher or lower **init** state to **init** state 2, files that begin with an "S", "P", or "I" are executed with the **start** option.
- When changing from a higher **init** state (3-9) to **init** state 2, scripts in *rc2.d* that begin with a "K" are executed with the **stop** option.
- Files that begin with characters other than "S", "P", "I", or "K" are ignored.
- Files are executed in order of their filenames sorted according to the ASCII collation sequence from their second character onwards (see **rc2**(ADM) and **prc_sync**(ADM) for more details). Scripts that start with "S" or "I" (or "K" for */etc/rc0.d* shutdown scripts) are executed serially; those beginning with "P" are executed concurrently if they form a contiguous sequence between "S" or "I" (or "K") scripts. Processing of "S" or "I" (or "K") scripts does not start until all previous scripts have exited.

  Table A-6 (this page) gives, in sort-sequence order (left to right), the ASCII characters that are valid for naming files.

Table A-6 Valid filename characters

#	%	+	,	-	.	0	1	2	3
4	5	6	7	8	9	:	=	.?	@
A	B	C	D	E	F	G	H	I	J
K	L	M	N	O	P	Q	R	S	T
U	V	W	X	Y	Z	^	_	a	b
c	d	e	f	g	h	i	j	k	l
m	n	o	p	q	r	s	t	u	v
w	x	y							

- Your script may rely on the existence of services or daemons (such as network services, the **cron** daemon, or the print scheduler) started by other scripts. For example, if your script depends upon certain filesystems being mounted, make sure that you name your script so that it runs after the **I01MOUNTFSYS** script. (When you add a new filesystem with **mkdev fs**, the appropriate files are updated with the information necessary to mount the new filesystem when the system enters multiuser mode.)

To add a function to the initialization procedure:

1. Write a script that performs the desired function.
2. Test the script to make certain it behaves as expected. Be sure any environment variables used in the script are defined at startup.

3. Name the file so that it begins with the uppercase letter "P", "S", "I", or "K" followed by a two-digit number indicating the order in which it should be executed relative to the other files in the directory, and ends with a name that describes the script's function. For example, **S80lp** handles print service startup. It will be executed after any script that begins with **S79**, and before any that begins with **S81**. You must follow this naming convention to ensure that your script is executed at the proper time.

   Note that a set of scripts whose names start with **P77**, **P78**, and **P79** will be executed concurrently. **S80lp** will not start until they have all exited.

4. Copy the script into the */etc/rc2.d* directory so that it is executed by **rc2** when the system enters (or leaves) multiuser mode.

If the function that you want to add is in the same category as functions performed by a script already located in */etc/rc2.d*, simply edit the existing script to add the new function. For example, you can add a function related to UUCP to the file *P70uucp*. You can also edit any script to tailor it to your needs. For example, to start process accounting, remove the appropriate comments from the *P16KERNINIT* file. Remember to back up the original script before modifying it.

Starting daemons on a trusted system

If your system is configured in "High" or "Improved" (C2) security mode, all processes must be stamped with an LUID (login user ID) in order to run properly. If you add any scripts that call a **setuid** or **setgid** (set user ID or group ID) program to the */etc/rc* structure, you must remember to set the LUID. This ensures that the system accurately records who does what, even if the effective identity of the user changes. An attempt to run a **setuid** or **setgid** program without an LUID fails with the error message: `cannot execute`. If the program does not change the ID of the process, this procedure is unnecessary.

To set the LUID, use the **su**(C) command:

su ***username*** **-c** "***command***"

where ***username*** is the name of the user or account and ***command*** is the complete command that you want to execute.

For example, the **deliver** daemon checks periodically for undelivered mail. The daemon is in the file */etc/rc2.d/S86mmdf*.

```
/bin/su mmdf -c "/usr/mmdf/bin/deliver -b"
```

This command sets the LUID to *mmdf*, which is the LUID used to administer the mail routing system. The command is run automatically when the system goes into multiuser mode.

Daemons that must run without an LUID

The **sdd** daemon process is used to start processes that must run without an LUID (such as **cron**). Processes that must have no LUID can be added to the */tcb/files/no_luid/cmdtable* file. See the **sd**(ADM) manual page for more information.

Modifying .profile and .login files

The */etc/profile* file sets the default environment for all users of the Bourne and Korn shells, while */etc/cshrc* does the same for users of the C shell. The *.profile* and *.login* (for C shell users) files in the users' home directories contain commands that initialize the environment for each individual user. When a Bourne or Korn shell user logs in, the shell first executes the commands in */etc/profile* and then executes the commands in the *.profile* file in the user's home directory. When a C shell user logs in, the shell executes the commands in */etc/cshrc* and then the commands in the user's *.login* file in their home directory. (Depending on the login shell, other files may also be read.)

The files contain commands that set various system variables (for example, **TERM**, **PATH**, and **MAIL**). These variables give the system information such as what terminal type is being used, where to look for programs that the user runs, where to look for the user's mailbox, and what keys to expect for the "kill" and "backspace" functions. For more information about these environment variables, see "Understanding variables" in the *Operating System User's Guide*.

There is one *.profile* and/or *.login* file for each user account on the system. The files are placed in the user's home directory when the account is created. Users can modify their own *.profile* or *.login* files or allow the system administrator to make modifications. In either case, these files are ordinary text files and can be modified using a text editor. Commands can be added or removed as desired.

Changing the /etc/motd file

The message of the day file, */etc/motd*, is intended to provide the greeting displayed whenever a user logs in. By default, this file is empty. */etc/motd* is an ordinary text file, so the system administrator can change the message by editing the file with a text editor. You can also use the **MOTD Manager** located in the *System* directory of the SCOadmin hierarchy, which simply invokes the default editor on the file.

In general, you should limit the size of the */etc/motd* file to include no more than a single screen of information. You can modify this file to include messages such as a reminder to clean up directories, to preserve disk space, a notice of the next periodic backup, a description of the latest system upgrade, or information about upcoming scheduled system down times.

Other message files

There are other message files on the system that are displayed at boot or login time:

/etc/issue	contains the SCO product name
/etc/copyrights	a directory containing copyright messages for components of your SCO system

Appendix B

Using the system console and non-graphical displays

This appendix describes utilities and features that affect the use of the system console and non-graphical displays, including serial terminals. Much of the information presented here concerns use of the console in non-graphical mode.

Console displays are connected to a standard display adapter, while color terminals are connected to the system by special adapters.

Tasks discussed here include:

- Using multiscreens (page 412)
- Using the console screen protection feature (page 413)
- Changing non-graphical video fonts (page 413)
- Controlling non-graphical color displays with setcolor (page 414)
- Setting the console keyboard type (page 416)
- Using serial multiscreens with mscreen (page 417)

If you want to set up a serial console, refer to "Setting up serial consoles" in the *SCO OpenServer Handbook*.

Using multiscreens

With multiscreens, you can use your console as several terminals at one time. Use of this feature is described in "Running programs simultaneously with multiscreen displays" in the *SCO OpenServer Handbook*.

See also:

- **multiscreen**(M)
- **screen**(HW)

Reducing the number of multiscreens

The system is configured with 12 **multiscreen**(M) console screens by default. Although this does not significantly affect performance, you can reduce the number of screens if desired. To configure the system for fewer screens, do the following:

1. Log in as *root* and use **configure**(ADM) or the **Hardware/Kernel Manager**. Select category 9, "TTY and console configuration."
2. Skip the parameters displayed by pressing ⟨Enter⟩ until you reach the **NSCRN** parameter. Enter a value corresponding to the number of screens you wish to enable.
3. Calculate the amount of screen memory in KB (controlled by the **SCRNMEM** parameter) as follows:

   For 25-line displays: **SCRNMEM** = 10KB + 4KB* **NSCRN**

   For 43-line displays: **SCRNMEM** = 10KB + 8KB * **NSCRN**

4. Enter a value for **SCRNMEM** to match the value you calculated in the previous step.
5. Exit the **configure**(ADM) menu by entering " q " and pressing ⟨Enter⟩.
6. Relink the kernel as described in "Relinking the kernel" in the *SCO OpenServer Handbook*.
7. Disable the unused multiscreen ttys. For example, if you reconfigured for 5 screens after having 12, you would enter the following command at the system prompt to disable ttys 6 through 12:

   disable tty06 tty07 tty08 tty09 tty10 tty11 tty12

8. Shut down the system and reboot from the new kernel. Use the **System Shutdown Manager** or this command:

   shutdown -g0

   You can warn users by replacing 0 with the number of minutes. When the reboot message is displayed, press ⟨Enter⟩ to restart the system.

Multiscreens and multiple video adapters

Video adapters can be assigned to multiscreens dynamically. All start on the primary adapter, but any screen can be moved to another video card with the **vidi**(C) command.

Valid adapter names are "mono", "cga", "ega", and "vga".

For example, if your primary video adapter is an EGA and you have a MONO secondary adapter, you can move the current screen onto the MONO card using the following command:

```
vidi mono
```

Using the console screen protection feature

VGA consoles can be set up to blank after a certain number of seconds to protect the screen from excessive wear. The kernel parameter **TBLNK** controls the VGA console screen protection feature. By default, screen blanking is not performed: to enable this feature, log in as *root* and use the **configure**(ADM) command or the **Hardware/Kernel Manager** described in "Configuration tools" in the *Performance Guide*. Select category 9, "TTY and console configuration" and change the value of **TBLNK** to the number of seconds that the system should wait before blanking the screen. Then relink the kernel as described in "Relinking the kernel" in the *SCO OpenServer Handbook*.

Changing non-graphical video fonts

You can display the full range of characters on a display adapter by using the **vidi**(C) utility. Normally, if you have a console with a display adapter which has a character set defined in a ROM, you will be able to display only those characters defined in that ROM. In addition, in order to display the entire font set, the **mapchan** file for the console must correspond to the character set defined in that ROM.

In addition to using **vidi**(C) to override ROM, you can use it to define certain display fonts on some display adapters. For example, the VGA adapter will allow you to display fonts in the sizes 8x8, 8x14, and 8x16.

The **vidi**(C) utility defines the font for one of these six character sets.

Table B-1 Font definition files (*/usr/lib/vidi*)

Character set	8x8 font	8x14 font	8x16 font
PC standard	font8x8	font8x14	font8x16
ISO 8859/1	iso.8x8	iso.8x14	iso.8x16
PC Nordic	nor.8x8	nor.8x14	nor.8x16
PC Portuguese	por.8x8	por.8x14	por.8x16
PC Spanish	spa.8x8	spa.8x14	spa.8x16
PC Greek	grk.8x8	grk.8x14	grk.8x16

Controlling non-graphical color displays with setcolor

setcolor(C) is a simple utility that enables you to control the colors used on the display screen. Both foreground and background colors can be set independently in a range of 16 colors. **setcolor** can also set the reverse video and graphics character colors. (The **setcolor** command usually has no effect on monochrome displays or terminals.)

NOTE **setcolor** also works on the ansi display within a **scoterm**(XC) window.

The following colors are available:

blue	magenta	brown	black
lt_blue	lt_magenta	yellow	gray
cyan	white	green	red
lt_cyan	hi_white	lt_green	lt_red

To display these colors, simply invoke **setcolor** without options.

The sections that follow explain the options for using these colors.

Changing the foreground and background colors

You can set both background and foreground colors with a single command:

setcolor red white

This results in red characters on a white background. If only one color is specified, the foreground color is changed. To change only the background color, use the **-b** option, as in the following:

setcolor -b red

This changes the background color to red.

Changing reverse video colors

setcolor allows you to set the reverse video colors for the background and foreground independently. For example:

```
setcolor -r blue red
```

This command sets the foreground reverse video color to blue and the background reverse video color to red.

Changing the screen border color

For CGA cards you can also change the color of the square border that defines the text region of the display:

```
setcolor -o green
```

The above example changes the border to green without affecting the rest of the display.

Sounding the keyboard bell

One of the less obvious functions of **setcolor** is to control the sound of the bell that is usually built into the display or keyboard. To change the bell tone, you must supply a pitch and duration. Pitch is the period in microseconds, and duration is measured in fifths of a second. When using this option, a ⟨Ctrl⟩**G** (bell) must be echoed to the screen for the command to work. For example:

```
setcolor -p 500 2
echo ^G
```

This command sets the bell to a high pitch of short duration. The higher the pitch number, the lower the sound generated. This command sets the bell to a sustained low tone:

```
setcolor -p 7000 8
```

Note that each time ⟨Ctrl⟩**G** is pressed, the bell will sound the tone most recently set.

Resetting the screen

setcolor -n returns the screen to the default white characters on black background.

Setting the console keyboard type

SCO systems support two keyboard modes: AT and XT. By default, the system is configured for use with an XT keyboard. This is because an XT (or other non-AT) keyboard will not work in AT mode. The system will not recognize keyboard input. An AT keyboard will work properly in XT mode, but the extended keyset found on the AT 101 or 102 key keyboard is not accessible. If you have an AT keyboard you can reset the keyboard mode to AT to make full use of the extended keyset. Use the **kbmode**(ADM) utility is used to test and set the keyboard mode.

CAUTION Do not change the keyboard type to AT if you are running graphical sessions on the console. The X server does not support it.

Some keyboards have an AT keyboard layout, but do not support AT mode. To test your keyboard to determine if it supports AT mode, invoke **kbmode** with the test option:

kbmode test

A sample session with **kbmode** in test mode is shown below, with user input in bold:

```
# kbmode test
Current keyboard mode is XT
Do you want to determine if your keyboard supports AT mode? y
During the test the keyboard will be put into AT mode.
You should then press the space bar two or three times.
Are you ready to start? y
Please hit the space bar now!

The keyboard has been returned to its default mode.
It supports AT mode.
#
```

The display will be temporary initialized to AT mode.

Switching keyboard modes manually

The **kbmode** utility is also used to set the mode. Use one of the following commands for switching to AT or XT mode:

kbmode at
kbmode xt

Changing modes permanently

To change the default mode permanently, the kernel parameter **KBTYPE** must be set to the proper keyboard. To change **KBTYPE**, log in as *root* and use the **configure**(ADM) command or the **Hardware/Kernel Manager** described in

"Configuration tools" in the *Performance Guide*. Select category 18, "Miscellaneous device drivers and hardware parameters" and change the value of **KBTYPE**. Then relink the kernel as described in "Relinking the kernel" in the *SCO OpenServer Handbook*.

Using serial multiscreens with mscreen

The **multiscreen**(M) feature provides many separate login screens on the console. It is possible to use a similar feature on a terminal. Terminals that have multiple pages of screen memory can be used as separate screens, each with a different login session, as if you had several terminals at your service instead of one.

On a Wyse 60 terminal, the contents of two entire screens of activity can easily be saved. The use of a third screen on the Wyse 60 is discussed below. Using two screens is very much like having more than one terminal. The complete functionality of a login session is provided on each screen, and previously executed commands (or their results) are displayed on each screen when it is in use. This section uses the Wyse 60 as an example. (See the **mscreen**(M) manual page for a technical explanation.)

You can also limit the number of mscreens available. The **mscreen** utility provides access to multiple terminal sessions, much like logging in on more than one terminal. These sessions are provided on "pseudo-ttys" rather than the tty devices usually used by terminals or modems. A tty is a special file associated directly with a particular hardware device used for communication with equipment such as terminals or printers. ttys can be seen in the */dev* directory as files with the name "tty" followed by a number and a letter. Use the **mkdev pttys** command to add pseudo-ttys for use with **mscreen**. As a general guideline, the recommended number is two or three per user — so if you plan to accommodate 8 serial terminal users, you should create 16-24 pseudo-ttys. Refer to "Adding or removing pseudo-ttys" in the *Networking Guide* for more information.

Adding more **mscreen** capability to your system should increase the productivity of the users. However, too much of a good thing can slow your system down. A system with 10 users, each of whom uses two screens, could perform as though it is servicing 20 users. Keep system performance in mind when deciding how many mscreens should be allowed system-wide, and who should be able to use them.

NOTE When using the **who** command, each user **mscreen** session is listed. If you wish to list only the master logins, use the **who -f** command.

No terminal known contains enough screen memory to save the material displayed during the use of all 20 logins that **mscreen** is capable of. However,

any terminal should allow the user to switch between as many as 20 screens, providing the keyboard has enough extra keys to indicate the switch between screens. The user will probably not find multiple screens very useful without multiple pages of screen memory. It is inconvenient, for example, to have to redraw the terminal's screen each time one switches screens when using a spreadsheet on one screen and **vi** on the other. Most people who use terminals with minimal screen memory prefer shell layers **shl**(C) to **mscreen** for multiple login sessions. For more information, see the **shl**(C) manual page.

Adding pseudo-ttys

A pseudo-tty is a device that is not associated with any real hardware, and it is used to simulate the function of a real tty. Users of networking products should already be familiar with pseudo-ttys, as they are the devices used to log in on remote machines. A pseudo-tty is represented by two software devices that appear in a listing of */dev* as *ptyp* and *ttyp*, each followed by a number. The former is called the "master" tty and the latter the "slave". Between the two, they simulate a functional tty.

Pseudo-ttys are created with the **mkdev ptty** command.

To configure the pseudo-ttys, log in as *root* and enter the following command:

mkdev ptty

This automatically creates the necessary devices, updates the files */etc/inittab* and */etc/conf/cf.d/init.base*, and updates the **NSPTTYS** kernel parameter (the maximum number of pseudo-ttys) as necessary. If this value is increased, a kernel relink will be necessary.

mscreen troubleshooting

Unlike many utilities, **mscreen**'s complex responsibilities require a number of conditions for correct functionality. By following the suggestions here, you should be able to avoid some of the more common mistakes made by new **mscreen** users.

In preparing to use **mscreen**, make sure your terminal works with the program. Find out how much screen memory is provided by consulting your terminal manual. The **mscreen** utility uses the file */etc/mscreencap* to determine how to change screen images for your particular terminal. As shipped, */etc/mscreencap* is supplied with only a few terminals. To use **mscreen** with other terminals, configure the */etc/mscreencap* file before using your terminal. If you run **mscreen** on a terminal that does not have an entry in */etc/mscreencap*, **mscreen** fails.

If you are sure your terminal works with **mscreen** and you have a working *mscreencap*, but **mscreen** still fails:

- Create more pseudo-ttys with **mkdev ptty**.

  You may need to create more pseudo-ttys if the pseudo-ttys currently on your system are in use.

- Verify switching.

  Make sure the */etc/mscreencap* for your terminal is correct. Use one of the examples in */etc/mscreencap* to check the way your function key output sequence is mapped to a particular **mscreen** command. You must log in separately to each screen you intend to use.

- Kill **mscreen** processes.

  If you are testing an *mscreencap* entry and you have trouble with the screens, you should do the following:

  1. Check the processes that are running:

     ps -u*username*

  2. Kill all the **mscreen** processes:

     kill -9 *process_numbers*

Advanced mscreen configuration

Many users find **mscreen** satisfactory as provided. For advanced **mscreen** users, or anyone interested in learning more about both **mscreen** and the operating system, here are some "tuning" tips for using and extending **mscreen**.

In addition to invoking **mscreen** automatically, the script in Example B-1, ".login script" (page 420) allows three full-featured mscreens on a Wyse 60 and adds a number of convenience features for the **mscreen** user. Example B-2, ".profile script" (page 420) presents the same material for the Bourne and Korn shell *.profile* file. These examples are designed to be added to the end of your *.login* or *.profile* file, and replace any existing **tset** material.

Example B-1 .login script

```
# Example material for the end of a C-Shell .login file.
#
# If logging in via pseudo-tty, suppress terminal initialization.
set ttyname=`tty`

# Set init to null, initially.
set init = ""

set noglob

# Reset init to the value "-I" when logging in on a pseudo-tty to
# suppress the tset terminal initializations string.

if ( `expr $ttyname : "/dev/ttyp"` > 0 ) set init = "-I"
set term = (`tset -m ansi:ansi -m wy60:wy60 -m:\?wy60 -r -S -Q $init`)
setenv TERM $term
unset noglob term
# Put WYSE 60 in ECON-80 mode during initial log in process.
if ( "$init" != "-I" && "$TERM" == "wy60" ) /bin/echo "\033eG\c"

# Set the prompt to indicate the tty number of the current
# mscreen and command.
set prompt = "`expr $ttyname : '/dev/\(.*\)'` \!% "

# Release the local variables used.
unset ttyname init

# Run mscreen and logout if the 'stop' key (defined as S-F9 in
# the default /etc/mscreencap for wy60) is pressed. This string
# is described in the mscreen (M) manual pages.
mscreen -n 3
if($status == 0) logout
```

Example B-2 .profile script

```
# Example material for the end of a Bourne shell .profile file.
#
ttyname=`tty`

init=""

if [ `expr $ttyname : "/dev/ttyp"` -gt "0" ] ; then
        init="-I"
fi
eval `tset -m ansi:ansi -m wy60:wy60 -m :\?wy60 -r -s -Q $init`
export PATH

if [ "$init" = "-I" -a "$TERM" = "wy60" ] ; then
        /bin/echo "\033eG\c"
fi

PS1="`expr $ttyname : '/dev/\(.*\)'` $ "

unset ttyname init

mscreen -n 3
if [ "$?" = "0" ] ; then
        exit
fi
```

Many *termcap* entries (including wy60) clear the screen buffers (which **mscreen** uses to store the contents of multiple screens) as part of the initialization string. In Example B-1, ".login script" (page 420) and Example B-2, ".profile script" (page 420), **tset**(C) sends the initialization string only during the first login procedure. When logging in on pseudo-ttys, **tset** is invoked with the **-I** flag. This is done by adding the **init** variable to the **tset** line. The first time **tset** is run, **init** has a value equal to "", adding nothing to the **tset** command. When it is run subsequently, **init** has a value of "-I", adding the option to **tset**.

Following the **tset** command, the string "\033eG\c" is echoed during first login procedure. This escape sequence changes the "COLUMNS" setting in the Wyse 60 to ECON-80 mode. The combination of these settings frees up just enough screen memory to use three screens. As an extra convenience, the user's prompt is set to display the current slave pseudo-tty number, allowing the user to keep track of which screen is in use.

If you do not use a Wyse 60 terminal, you can still set your prompt to indicate the current screen, and invoke **mscreen** automatically while checking for the shell return code, as illustrated in Example B-1, ".login script" (page 420) and Example B-2, ".profile script" (page 420).

Appendix C

UNIX directories and special device files

This appendix lists the most frequently used files and directories on a UNIX system. Many of these files and directories are required for proper system operation and must not be removed or modified. The following sections briefly describe each directory.

This appendix also describes device nodes relating to filesystems, terminals, and other devices. For a full description of the special files mentioned here, see the manual pages in the HW section.

NOTE Most system files are symbolic links to locations in the */opt* and */var* directories. See "The /opt directory" (page 426).

The root directory

The root directory (/) contains these system directories:

/bin UNIX command directory (page 424)

/dev device special directory (page 424)

/etc additional program and data file directory (page 425)

/lib C programming library directory (page 426)

/mnt mount directory — reserved for mounted filesystems (page 426)

/opt location of shared software storage object files (page 426)

/shlib shared libraries (page 426)

/stand filesystem containing the kernel and boot files (page 427)

/tcb TCB (Trusted Computing Base) security binaries and databases (page 427)

/tmp temporary directory (reserved for temporary files created by programs) (page 428)

/usr user service routines — may contain user home directories (page 427)

/var location of non-shared software storage object files (page 428)

The /bin directory

The */bin* directory contains the most common UNIX commands — the commands likely to be used by anyone on the system.

The /dev directory

The */dev* directory contains special device files that control access to peripheral devices.

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 system 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*.

*/dev/vdisk** devices are associated with the **Virtual Disk Manager** (page 355).

Table C-1 Commonly-used /dev device nodes

UNIX device	XENIX device	Name
/dev/console	-	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/vdisk0	-	virtual disk configuration database
/dev/vdisk1	-	root filesystem mirror
/dev/vdisk2	-	swap area mirror
/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	-	lineprinter
/dev/kmem	-	kernel virtual memory
/dev/mem	-	physical memory
/dev/null	-	null device
/dev/rmt0	/dev/rct0	default cartridge tape device
–	/dev/rft0	QIC-40 tape device
–	/dev/rctmini	minicartridge tape device
/dev/root	-	root file structure
/dev/swap	-	swap area
/dev/ptyp*nn*	–	pseudo-tty (master)
/dev/ttyp*nn*	–	pseudo-tty (slave)
/dev/tty*nn*	–	console multiscreen
/dev/tty1a	-	main serial line
/dev/tty2a	-	alternate serial line
/dev/tty1A	-	main serial line (modem control)
/dev/tty2A	-	alternate serial line (modem control)

The /etc directory

The */etc* directory contains miscellaneous system program and data files. 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. See "Changing scripts in /etc/rc2.d" (page 405) for more information on the */etc/rc* directories.

The link kit (kernel configuration files, device drivers, and associated utilities) is located in the */etc/conf* directory. See the **configure**(ADM), **mkdev**(ADM), **mtune**(F), **stune**(F) manual pages and the *Kernel/Driver Interface*.

The data files in the directory */etc/default* contain default information used by system commands. Each file contains information explaining its purpose and the associated command or application. See "Using the System Defaults Manager" in the *SCO OpenServer Handbook* or the **default**(F) manual page for more information.

The /lib directory

The */lib* directory contains runtime library files for C and other language programs. This directory is required.

The /mnt directory

The */mnt* directory is an empty directory reserved for mounting removable filesystems.

The /opt directory

The */opt* directory contains the actual system software in a series of subdirectories known as "software storage objects" or SSOs. All system utilities in the root filesystem (such as */usr*, */bin*, and so on) are actually links to an SSO. */opt* contains the shared files of SSOs that can be used by clients of a server system, while */var* contains the non-shared files. See "Software storage objects" in the *Networking Guide* or the **hierarchy**(M) manual page for more information.

The /shlib directory

The */shlib* directory contains shared libraries used by the system. These libraries are called by system utilities at run time instead of compiled into the actual binaries.

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 used less frequently or are 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* and */usr/adm/syslog* files contain a record of system error messages, including those sent to the system console. These files are especially useful for locating system problems, including hardware problems. For example, an unusual number of disk errors on a drive indicates a defective or misaligned drive. Because messages in the file can accumulate rapidly, these files must be deleted periodically. See "Checking and clearing system log files" (page 91) for more information.

The /stand directory

The */stand* directory is the mount point for the boot filesystem, which contains the loader (*boot*), the kernel (*unix*), and associated boot files (including *bootos*, which loads operating systems other than UNIX).

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. The TCB database files and their formats are discussed in the **authcap**(F) manual page. For a complete discussion of system security, see Chapter 5, "Maintaining system security" (page 223).

The /tmp directory

The */tmp* directory contains temporary files created by UNIX system 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 can remove any temporary file that does not belong to a running program.

The /var directory

The */var* directory contains the non-shared SSO files that are specific to an individual client or server. See "The /opt directory" (page 426) for more information.

Appendix D

Using the crash(ADM) diagnostic tool

The **crash**(ADM) command is an advanced system diagnostic tool for use by experienced system administrators. It allows you to examine the memory image of a live or crashed system. It has a large number of internal commands that format and display kernel control and data structures, system tables, and other information. It will also evaluate expressions given as arguments to these commands. These expressions are written in a syntax similar to that of the C programming language.

See also:

- the **crash**(ADM) manual page
- "Running the crash command" (this page)
- "Examples of using crash" (page 431)

Running the crash command

To run **crash** to examine a live system, log in as *root* and enter the command:

/etc/crash

crash reads the kernel program file for symbol table information. It obtains the system memory image by opening the special device file */dev/mem*. If the currently booted kernel is not */unix* (symbolically linked to */stand/unix*), you must specify the pathname of the kernel program file using the **-n** option. For example, if the kernel running at the time of the crash was */stand/unix.old*, you would enter:

/etc/crash -n /stand/unix.old

To examine a memory image that you have saved to a file using **ldsysdump**(ADM), you must specify the name of this file using the **-d** option.

For example, if you have used **ldsysdump** to save the memory image to the file */tmp/dumpfile*, you would enter:

/etc/crash -d /tmp/dumpfile

Again, if the currently executing kernel is not the same as that running when the system crashed, you must specify the name of the correct kernel program file to **crash** using the **-n** option. For example, if you renamed the program file for the kernel that crashed as */stand/unix.crashed*, enter:

/etc/crash -d /tmp/dumpfile -n /stand/unix.crashed

See "Examples of using crash" (page 431) for typical instances in which you might use **crash**.

Defining the default dump device

If your system crashes, it writes its memory image to the dump device. The default dump device is the swap area, */dev/swap*, but you can define an alternative dump device in one of two ways:

- by using the **dump** bootstring at the `Boot:` prompt, or by including this as part of the default bootstring in the **boot** defaults file, */etc/default/boot*; see the **bootstring**(HW) and *boot*(F) manual pages for more information. You can specify a tape drive as the dump device. For example:

  dump=Stp(8)
  dump=ct(8)

  define the dump device to be a SCSI tape drive or a cartridge tape drive respectively. This can be useful if the size of the swap area is too small to hold the system image — if, for example, you have configured a swap area smaller than the amount of physical memory on your system.

  If you do not want the system to save its memory image, specify:

  dump=none

- by redefining the system default dump device in the file */etc/conf/cf.d/sassign*; this definition will be included in the kernel when you next rebuild it. Using *sassign*(F) in this way, you can also define a tape drive as the dump device. For example:

  dump Stp 8
  dump ct 8

  define the dump device to be a SCSI tape drive or a cartridge tape drive, respectively.

  Note that you must edit *sassign*(F) and then relink the kernel to redefine the dump device.

If your system dumps its image to the swap area, it runs the **dumpsave**(ADM) command when it is next booted. This allows you to save the memory image to tape or floppy disk for later analysis.

NOTE If you intend to analyze a memory image on a different system than the one that crashed, you must make a copy of the kernel program file that was running on the system that crashed. You cannot examine a memory image without this file — **crash** needs it to access its symbol table.

If you also need to find out which files were open to processes at the time of the crash, you will also need to copy the contents of the filesystems.

The complementary command **ldsysdump**(ADM) copies a memory image from tape or floppy disk to a file:

/etc/ldsysdump *filename*

To check the validity of a memory image, you can use the **memsize**(ADM) command:

/etc/memsize *filename*

memsize returns an error if the memory image is corrupted.

Examples of using crash

The following sections walk you through some examples of using **crash** to examine a live system or a dump from a crashed system:

See also:

Examining processes

The **crash** command provides several commands for examining a process:

Examining the process table

The **proc** command of **crash** displays information held in the kernel's process table. It shows process information similar to that displayed by the **-elf** options of the **ps**(C) command. The following example shows **proc** being used to display the first eight slots of the process table:

```
> proc 0..7
PROC TABLE SIZE = 47
SLOT ST PID PPID PGRP UID PRI CPU EVENT           NAME        FLAGS
   0 s   0    0    0   0  95   0 runout          sched       load sys lock nwak
   1 s   1    0    0   0  66   1 u               init        load
   2 s   2    0    0   0  95   0 kspt1+0x10bbdc  vhand       load sys lock nwak nxec
   3 s   3    0    0   0  81   0 kspt1+0xfe64c   bdflush     load sys lock nwak nxec
   4 s   4    1    1   0  95   0 vm_dma_end+0x11cc8 kmadaemon load sys lock nwak nxec
   5 s   5    1   11   0  95   0 kspt1+0xcbdc8   htepi_daemon load sys lock nwak nxec
   6 s   6    1   16   0  95  22 pbintrpool     strd        load sys lock nwak nxec
   7 s 289    1  289   0  73   0 proc+0x968      ksh         load
```

The kernel daemons **sched** (swapper), **init** (process spawner), **vhand** (page handler), **bdflush** (buffer flusher), **kmadaemon** (kernel memory allocator), always occupy slots 0 through 4. **htepi_daemon** (HTFS filesystem daemon), and **strd** (STREAMS daemon) may not be present on your system.

Output columns of interest include:

`ST` Shows the process status. These are lowercase equivalents of the process state indicators documented on the **ps**(C) manual page. For example, `s` means the process is paused or sleeping on some resource, `r` represents a process that is ready to run, and `o` is a process that is running on a CPU.

PRI Shows the priority of the process. Note that on configurations that use the fixed priority scheduler, 127 is the highest priority and 0 is the lowest priority. User mode processes have priorities from 0 to 65, system mode processes have priorities from 66 to 95, and fixed priority schedule process priorities range from 96 to 127. (Priority 51 is the default priority for user-level processes.)

EVENT
Shows the event (symbol name and offset, or an address) on which a process is paused or sleeping.

FLAGS
Shows the process flags. For definitions of these flags, see the listing in the *<sys/proc.h>* header file. Note that **crash** displays the flags as all lowercase; it also removes the initial "S" and some additional letters. For example, the flag shown by **crash** as `nxec` and `nwak` are defined as **SNEXEC** and **SNWAKE** in *<sys/proc.h>*.

Examining the u-area of a process

In addition to the information about a process held in the process table, the user area ("u-area") of each process contains information that the process uses when it is running. The process table is permanently resident in memory, but a process's u-area can be swapped out to disk. The form of a u-area corresponds to the **user** structure defined in *<sys/user.h>*.

When issued without arguments, the output of the **user** command contains information about the u-area of the process that is currently executing or, for a memory image, the process that was executing at the time of the panic. If the system was executing the idle loop (no process was runnable) at the time of the panic, there is no current process.

You can specify a process table slot number (obtained from the **proc** command) to **user** to view the u-area for that process. You can also specify a process ID (PID) as an argument if you prefix it with the # unary operator.

The following example shows selected portions of the full display obtained using the **-f** option:

```
> user -f 86
PER PROCESS USER AREA FOR PROCESS 86
USER ID's:        uid: 13297, gid: 1014, real uid: 13297, real gid: 1014
        supplementary gids: 1014 50
PROCESS TIMES:    user: 19, sys: 131, child user: 6085, child sys: 7785
PROCESS MISC:
        command: ksh, psargs: -ksh
        proc: P#86, cntrl tty:  58,6
        start: Fri Jul 15 15:23:21 1994
        mem: 1e5, type: exec su-user
        proc/text lock: none
        current directory: I#360
OPEN FILES AND POFILE FLAGS:
           [ 0]: F#216   r    [ 1]: F#216     w  [ 2]: F#216     w
           [ 3]: F#292   r    [ 4]: F#174        [ 6]: F#156
           [31]: F#246 c r w
FILE I/O:
        u_base: 0x45164c, file offset: 10302696, bytes: 1230
        segment: data, cmask: 0022, ulimit: 2097151
        file mode(s): read
SIGNAL DISPOSITION:
        sig#      signal oldmask sigmask
           1:     0x6ffc    -     1
           2:     0x7718    -     2
...
```

The following sections are of interest:

`USER ID's`
: Shows the real and effective IDs of the user running the process.

`PROCESS MISC`
: Shows miscellaneous information about the process. The fields `command` and `psargs` tell you the name of the program and the first few arguments to the command. The `start` time shows the actual clock time when the process was initialized.

`OPEN FILES AND POFILE FLAGS`
: Tells you the files that the process had open, and the file descriptors involved. The file descriptors in use by the program are in shown square brackets; the `F#` numbers represent slot numbers in the open file table and can be used as arguments to the **file** command. For example, file descriptor 1 usually corresponds to the standard output (*stdout*) unless this was redefined. In this example, it points to slot 216 of the open file table; you can use the command **file 216** to view this entry.

SIGNAL DISPOSITION
Shows the behavior defined for all signals. See the **signal**(S) manual page for a definition of the signal numbers.

Finding out which files a process has open

It is sometimes useful to be able to find out which files a process has open. This may be because a program will run on one system but not on another. You may suspect that the permissions on a file or directory are preventing the program from running but you are uncertain as to which files it uses. Figure D-1 (this page) shows how entries in the file descriptor array (`u_ofile[]`) in each process's u-area point to entries in the open file table. Each entry in the open file table points to an entry in the generic in-core inode table. Note that a slot number in the in-core inode table is not the same (unless coincidentally) as the inode number of the associated file in the filesystem.

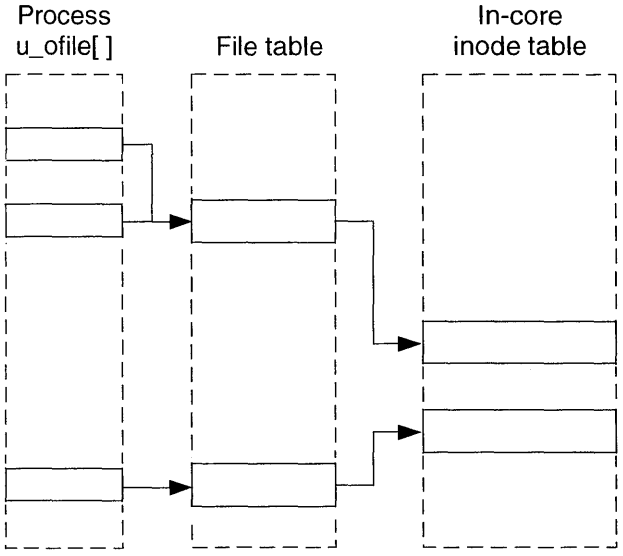


Figure D-1 The relationship between the file descriptor table of a process and the system open file and in-core inode tables

To find which files are open to a process, you must trace a path from the file descriptor table of a process to the in-core inode table. You can then use the **nodnm**(ADM) and **ncheck**(ADM) commands to find the filesystem to which the inode belongs and to relate one or more filenames to the inode.

1. Use the **proc** command to find the slot number of the process in which you are interested. On a live system, you can also use the **ps**(C) command to discover the process' ID number (PID).
2. Use the **curproc** command to define the process as the current process:

   ```
   > curproc slot
   Procslot = slot
   ```

   or

   ```
   > curproc #PID
   Procslot = slot
   ```

3. Use the **user** command to show which entries in the system file table are pointed to by the process' file descriptors:

   ```
   > user
   PER PROCESS USER AREA FOR PROCESS slot
   ...
   OPEN FILES AND POFILE FLAGS:
             [ 0]: F#3          [ 1]: F#10      w  [ 2]: F#10      w
             [ 3]: F#7       r
   ...
   ```

   The standard input (file descriptor 0) references entry 3 in the file table. Both the standard output and standard error output have opened entry 10 (file descriptor 2 is a duplicate of file descriptor 1), and file descriptor 3 references entry 7. The `r` and `w` flags show if the process has the files open for reading or writing, or both; the `c` flag shows a file descriptor that will be closed on an **exec**(S) system call.
4. Use the **file** command to find the in-core inode numbers for file table entries 3, 10, and 7:

   ```
   > file 3 10 7
   FILE TABLE SIZE = 341
   SLOT  RCNT   I/FL      OFFSET  FLAGS
      3     1 I# 267           0 read
     10     4 I# 291          94 write append
      7     1 I#   2           0 read write
   ```

5. Use the **inode** command to find the inode number and filesystem major and minor device numbers corresponding to each in-core inode number:

```
> inode 267 291 2
INODE TABLE SIZE = 161
SLOT MAJ/MIN FS INUMB RCNT LINK   UID   GID     SIZE    MODE MNT M/ST FLAGS
 267   1,40   1    45    7    1     0     3        0 c---666   0 S  0  -
 291   1,40   1    46    1    3     2    15        0 c---600   0 S  0  -
   2   1,40   1   171    1    1     9    16        0 p---060   0 S  0  -
```

From this, it can be seen that the in-core inode slot 267 refers to inode number 45 in the filesystem whose device node has major and minor device numbers 1 and 40. Other information that may be of interest is that inode number 46 has three hard links that reference it (LINK = 3), and inode number 45 has seven references to it in the file table (RCNT = 7).

The modes for both inodes 45 and 46 begin with "c"; this shows that the files are character special devices. Inode 171's mode begins with a "p" indicating that it is a named pipe. A regular file's mode would begin with a "f"; see the **ls**(C) manual page for a list of other mode character codes.

6. You can find the filesystem corresponding to the displayed major and minor device numbers using **nodnm**(ADM):

```
> !/etc/nodnm b 1 40
/dev/root
> !/etc/nodnm c 1 40
/dev/rroot
```

This shows that the filesystem that corresponds to device major number 1 and minor number 40 is */dev/root*.

7. The names corresponding to inodes 45, 46, and 171 in */dev/root* can be found using the **ncheck**(ADM) command:

```
> !ncheck -i 45 46 171 /dev/root
/dev/root:
45    /dev/null
46    /dev/console
46    /dev/syscon
46    /dev/systty
171   /usr/lib/cron/FIFO
```

From the above information, we can conclude that the process has opened the following files:

- */dev/null* on the standard input (file descriptor 0).
- */dev/console* on the standard output and standard error output (file descriptors 1 and 2). The standard error output descriptor is a duplicate of that for the standard output; both descriptors share the same open file, file pointer, and access modes. If the standard error output had opened this file separately, it would have its own entry in the file table rather than sharing one with the standard output.
- */usr/lib/cron/FIFO* on file descriptor 3.

Determining the size of a process

Each process has a per-process region (**pregion**) table associated with it that points to the regions that are private to it or that it shares with other processes. Figure D-2 (this page) shows how a process's pregion entry points into the system region table. This entry in the region table may be shared with other processes if it refers to program text (executable code).

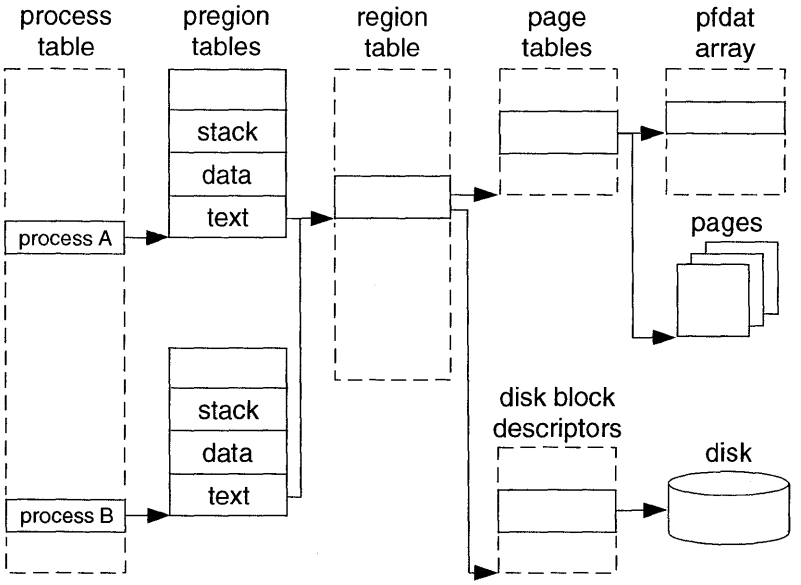


Figure D-2 The relationship between a process' pregion table and the system region table

To determine the current memory utilization of a process, you must add together the number of pages of physical memory that are described by the process' regions. If the size of several processes is being determined, you need to take care not to count shared regions more than once.

Use the **proc** and **region** commands of **crash**(ADM) to determine the size of a process as follows:

1. Issue the **proc** command to find the process table slot number for the program in which you are interested. This yields output similar to the following:

```
SLOT ST PID   PPID  PGRP UID PRI CPU EVENT        NAME        FLAGS
  11 s   17    12    12   0  33   0 proc+0x968   smallapp    load nxec
```

   The number in the first column (SLOT) is the process table slot number. The process ID is shown under the column PID. In this example, the program **smallapp** has process ID 17 and occupies slot 11 in the process table.

2. Run the **pregion** command to determine the regions associated with the process being studied. For example, enter **pregion 11** for slot number 11. The output has the following form:

```
PREG REG#       REGVA  TYPE FLAGS
   0   12           0  text rdonly
   1   20   0x400000  data
   2   21 0x7ffffffc stack
```

   The REG# column of this output shows the region table entries associated with this process; REGVA shows the virtual address of this region. TYPE identifies whether this region is text, data, stack, library text, library data, and so on; FLAGS identifies the characteristics of the region (read-access, write access, and so on).

3. Issue the **reg** command with the region table number(s) of the process. For example:

```
> reg 12 20 21

SLOT  PGSZ VALID  SMEM NONE SOFF  REF SWP NSW FORW BACK INOX TYPE FLAGS
  12     1     1     1    0    0   13   0   0   14    8    35 stxt done
  20     1     1     1    0    0    1   0   0  267   21    35 priv done
  21     2     1     1    0    0    1   0   0  267   22       priv stack
```

   The PGSZ column represents the maximum size of the region in pages. The VALID column indicates the number of 4KB pages that are currently in RAM. The TYPE column shows whether this is a private region (priv) or a shared text region (stxt).

   Summing the #VL sizes of all regions for the process shows the number of pages of memory being used by a program if it is only invoked once on the system. In this case, the sum is 3 pages. If multiple invocations of a program are run on a system, you can sum the #VL sizes for their shared text

and private regions separately; the same amount of memory will be dedicated to shared text regions regardless of how many invocations of that program are running. The sum of the #VL sizes for private regions is the amount of extra memory that will be consumed by each subsequent invocation of this program.

Finding regions shared by processes

You can use the **rtop** command of **crash** to find which regions are shared by processes. For example to determine which processes share regions 0 through 4, issue the command **rtop 0..4**.

Finding the largest processes on a system

Use the **region** and **rtop** commands of **crash** to identify the largest processes on the system.

1. To list the slot numbers of the five largest regions, enter:

```
> reg | sort +1rn | head -5 | awk '{print $1}'
12
5
8
0
1
```

The example output from this command shows the five largest regions in decreasing order of size.

2. Run **rtop** for the regions listed in the previous step to see which processes are associated with these regions. For the example above, you would enter:

```
> rtop 12 5 8 0 1
```

The output from this command will show the process table entries associated with each of the specified regions.

Examining kernel text

In this example, we examine the trap handler in the kernel that deals with page faults (trap type 14, or 0xe in hexadecimal; see the **trap**(M) manual page for a list of possible CPU exceptions).

1. Use the **idt** command to find the segment selector and offset of the trap handler from the Interrupt Descriptor Table (IDT):

```
> idt 0xe
iAPX386 IDT
CPU SLOT     SELECTOR OFFSET   TYPE       DPL  ACCESSBITS
  0   14         0158 f0011080 TGATE386     0  CNT=0
```

The displayed offset address, 0xf0011080 corresponds to the virtual address of the trap handler. The slot in the system GDT pointed to by the segment selector can be obtained by right-shifting its value by 3 places (0x0158 >> 3). This gives slot 43 in the GDT which describes the kernel's text segment. Figure D-3 (this page) illustrates how the offset point to the first level interrupt handler in the kernel's text segment.

2. The **dis** command can be used to examine the kernel code at this address. For example, to disassemble two lines of kernel code:

```
> dis 0xf0011080 2
pftrap              pushl  $0xe
pftrap+0x2          jmp    0xfffff0f1 <0xf0010178>   [cmntrap]
```

The handler routine **pftrap** calls the common trap handler routine **cmntrap**; this can be disassembled by specifying its symbolic name to **dis**:

```
> dis cmntrap 6
cmntrap             pushal
cmntrap+0x1         pushl  %ds
cmntrap+0x2         pushl  %es
cmntrap+0x3         pushl  %fs
cmntrap+0x4         pushl  %gs
cmntrap+0x5         pushfl
```

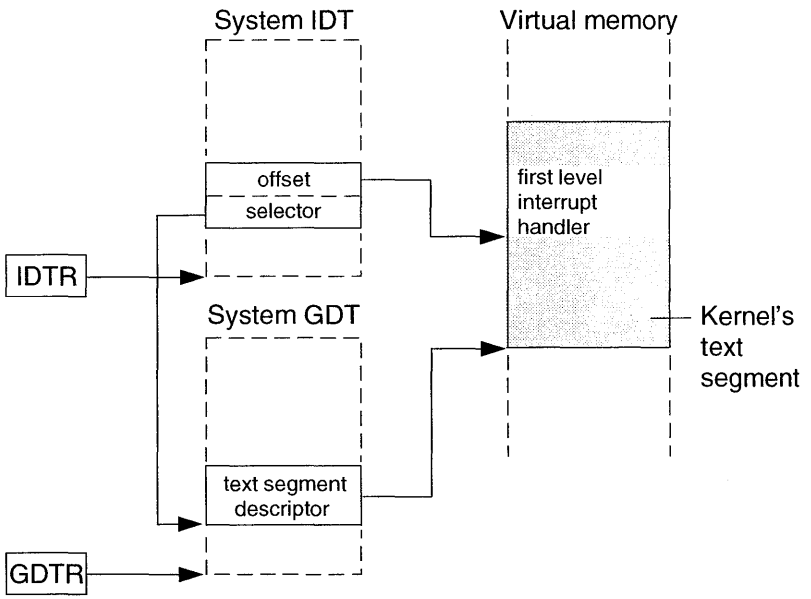


Figure D-3 How an Interrupt Descriptor Table entry indexes the first level interrupt handler in the kernel's text segment

NOTE On MPX systems, each active CPU has its own private IDT and GDT (there is, however, only one copy of the kernel's text.) If you do not specify a CPU to the **idt** or **gdt** commands, **crash** displays the descriptor tables for all CPUs. Use the **-c** ***cpu*** option to specify the CPU in which you are interested. The base processor is always ***cpu*** 0.

Studying a system panic

A system may crash if the kernel encounters an illegal condition from which it cannot recover. In such a case, it will "panic" and save its memory contents to the dump device before it stops running or "crashes". The default dump device is the swap area, */dev/swap*, but you can redefine this as described in "Defining the default dump device" (page 430). An example of when the system might panic would be if it encountered an uninitialized pointer in a third-party device driver. The memory image of a system that has crashed due to a system panic is also known variously as a "crash dump" or a "panic dump".

NOTE If the system was swapping when the panic occurred, the memory image may overwrite some of the pages being swapped. In such cases, not all the system information will be available to **crash**.

The memory dump saved after a panic contains information that can be used with any of the **crash** commands that are used on a live system as well as the **panic** and **trace** reports that give information about the state of the kernel when the system panicked. For more information about handling system crashes, see Chapter 13, "Troubleshooting system-level problems" in the *SCO OpenServer Handbook*.

panic command

In this example, we assume that the memory image exists on the swap device. The following command invokes **crash** specifying that it is to read the memory image from */dev/swap*, and use the symbol table in */stand/unix.test*:

crash -d /dev/swap -n /stand/unix.test

1. Use the **panic** command to discover the kernel trap type, the name of the routine and the address of the instruction that caused the panic:

```
> panic
System Messages:
. . .
ERR=0, TRAPNO=0
cs:eip=0158:f0055862 Flags=10286
. . .
eax= 00000001   ebx= f11100c0   ecx= 00000000   edx= f0055ac2
. . .
Kernel Stack before Trap:
STKADDR   FRAMEPTR  FUNCTION   POSSIBLE ARGUMENTS
e0000b94  e0000bc8  panioctl   (0xfacd0864,0x1,0,0x1)
. . .
```

   Trap number 0 shows that the panic was caused by a divide error. (See the **trap**(M) manual page for a list of possible CPU exceptions.)

   The first entry in the dump of the kernel stack shows the current stack frame when the panic occurred. This shows that the trap occurred within the routine **panioctl**.

2. Use the **dis** command to examine the kernel code at the address given by the value of the instruction pointer (`eip=0xf0055862`):

```
> dis 0xf0055862
panioctl+0xa5     idivl  %ecx,%eax
```

   This instruction performs an integer long division of the contents of register **EAX** by the contents of **ECX**. The contents of these registers shown in the output from **panic** are 1 and 0 respectively. We can conclude that the panic was caused by the attempt to divide by zero.

Note that this is a fictitious example; there is no such routine as **panioctl** in the kernel.

Examining a kernel stack trace

The **trace** command prints a symbolic stack traceback for any process or interrupt stack. When used on a postmortem memory image without options, **trace** prints the kernel stack that was present in the u-area of the user process that was executing at the time of the panic. **trace** also displays the contents of the processor registers.

You cannot use **trace** to examine the kernel stack of the current process on an live system. However, you can use it to examine the kernel stack of processes which are sleeping (for example, while waiting for I/O to complete).

Following an interrupt or exception, the kernel creates an additional stack frame on the system stack for the interrupt handler to use. If there was an interrupt outstanding on the crashed system being examined, **crash** also displays this interrupt stack and its register values.

Note the following when reading **trace** output:

1. The top two lines give the information from **crash proc** for this process.
2. The three bottom lines show the contents of the application program registers in the exception frame when the call was entered.
3. The lines above the register contents show the associated frame pointers, the associated function, and the pointer to the function argument(s). Note that **trace** does not differentiate between argument 0 and argument 1; both are reported as **ARG #1**.
4. If there is a second set of registers listed in the middle of the output, they are the registers from a second exception frame called during execution. Frequently this happens for an error exception, in which case **cmn_err** will appear in the **NAME** column to indicate a kernel error. The list of associated fram pointers are listed above the register contents.

Exception frames look very much like standard C stacks. **trace** searches for a pattern that could be an exception frame. Occasionally, **trace** will make a mistake. To verify that this is a real exception frame, verify that the frame pointers (**FP**) have numbers beginning with 4000, indicating that they are in the user area.

trace run on a postmortem dump displays the interrupt stack only if there were interrupts outstanding when the system crashed. If the interrupt stack was corrupted, using **trace -i** ***address*** enables you to get beyond the bad spot.

Determining the kernel component that failed

The information given by the **panic** and **trap** commands of the **crash**(ADM) command tells you which routines were executing just before the panic. Because the entire kernel (all kernel operating system code plus all driver code) is linked together into one executable file, this does not tell you which component caused the problem. Knowing which component contains the problematic function can sometimes tell you if the problematic function is part of the operating system, or if it is part of a driver supplied by another vendor.

This information can be extracted using the **strings**(C) command or, for systems where the SCO OpenServer Development System is installed, the **nm**(CP) command. Both methods are detailed below.

The following hints are provided for users who are not familiar with device driver conventions:

- If a routine is mentioned in a file named *Driver.o*, it is usually part of a device driver, and the driver prefix is the name of the directory in which the *Driver.o* file is located.
- If a function name ends in "read", "write", or "ioctl", it is part of a character device driver.
- If a function name ends in "strategy", it is part of a block device driver.
- Functions that end in "open" or "close" can be part of either a character or block device driver.
- Kernel utility functions are used in drivers much as system calls and library routines are used in user-level code. Section K of the manual pages (distributed with the *Device Driver Writer's Guide*) lists all the supported kernel utility functions. Some of the most frequently used functions are:

all drivers	**bcopy**(K), **cmn_err**(K), **copyin**(K), **copyout**(K), **inb**(K), **major**(K), **memget**(K), **minor**(K), **outb**(K), **sleep**(K), **spl**(K), **sptalloc**(K), **sptfree**(K), **timeout**(K), **wakeup**(K).
block drivers	**disksort**(K), **iodone**(K), **physio**(K).
STREAMS drivers	**allocb**(K_STR), **canput**(K_STR), **dupmsg**(K_STR), **flushq**(K_STR), **freeb**(K_STR), **freemsg**(K_STR), **getq**(K_STR), **linkb**(K_STR), **putnext**(K_STR), **putq**(K_STR), **qenable**(K_STR), **qreply**(K_STR),

Using strings(C) to find kernel component

To determine the file in which the problematic function is located using the **strings**(C)command:

1. Log in as *root*.
2. Determine the names of the kernel component files that reference the functions that were called just before the system panicked. The following shell script searches for the **sioopen**( ) function; you can type this in at the shell. Note that the "#" and ">" characters are prompts from the *root* shell; do not type them in. Be sure to get the single quote marks exactly right!

```
#for i in `find /etc/conf -name '*.o' -print`
> do
> string $i 2> /dev/null | grep sioopen > /dev/null && echo $i
> done
```

3. Write down the list of file names printed by the above command. For this example, you will get the output:

```
/etc/conf/pack.d/sio/Driver.o
```

4. Search for the file name in the */etc/perms* files. For this example, type:

```
egrep /etc/conf/pack.d/sio/Driver.o /etc/perms/*
```

   This will give as output:

```
/etc/perms/inst:LINK f644 root/sys 1 ./etc/conf/pack.d/sio/Driver.o N04
```

5. The `LINK` string after the colon indicates that this kernel routine is part of the link kit supplied as part of the operating system. If the file does not appear in the *perms* files, or if the package name that appears before the colon is part of a device driver not supplied by SCO, the problem may be with that driver.

Using nm(CP) to find kernel component

If the SCO OpenServer Development System software is installed on your machine, you can use **nm**(CP) instead of **string**(M) to extract this information. Use a shell script like the following to determine the file name that contains the routine, then proceed with steps 3-5 as shown in "Using strings(C) to find kernel component" (page 445).

```
# for i in 'find /etc/conf -name '*.o' -print'
> do
> nm $i 2 > /dev/null | grep sioopen > /dev/null && echo $i
> done
```

Additional help from SCO Support

If you are unable to determine the cause of a system panic, you may want to contact SCO Support for assistance. In addition to the system configuration information you filled out after installing the system, it is useful to have information from a memory dump available. To provide this:

1. Save the memory dump to tape and restore it to disk with the **ldsysdump**(ADM) command as discussed in "Recovering from a system panic" in the *SCO OpenServer Handbook.*

2. Extract the key troubleshooting reports from the dump with a command sequence similar to the following:

```
# crash -d /tmp/06May94 -w /tmp/crash.out
> panic
> trace
> user
> quit
```

   Remember that the " > " prompt is generated by the **crash** command, so do not type them in. This will create the */tmp/crash.out* file that contains the output from the **panic, trace,** and **user** functions from the **crash** command.

3. Print out the */tmp/crash.out* file and fax it to your support provider or e-mail the file itself.

Examining tty and cblock structures

The **tty** command prints the entries in the tty table for console multiscreens, parallel ports, serial ports, and pseudo-ttys. If you specify the **-f** option, the information displayed includes:

- the addresses of the control blocks and the first and last character block **cblock** structures in the character lists (**clist** structures) used to implement the raw input, canonical input, and output data queues
- control character settings
- line settings such as speed, parity, and character size

The **cblock** command allows you to examine the contents of the character blocks for a **clist** given its table entry displayed by the **tty** command. This may be useful if you wish to examine the a serial line which you think may be generating spurious characters. Figure D-4 (page 448) illustrates how a character list is built from a **clist** header and a chain of **cblock** elements. The **clist** header contains pointers to the first and last **cblock** only. The total count of characters in the c_data buffers is stored in the c_cc member of the **clist**.

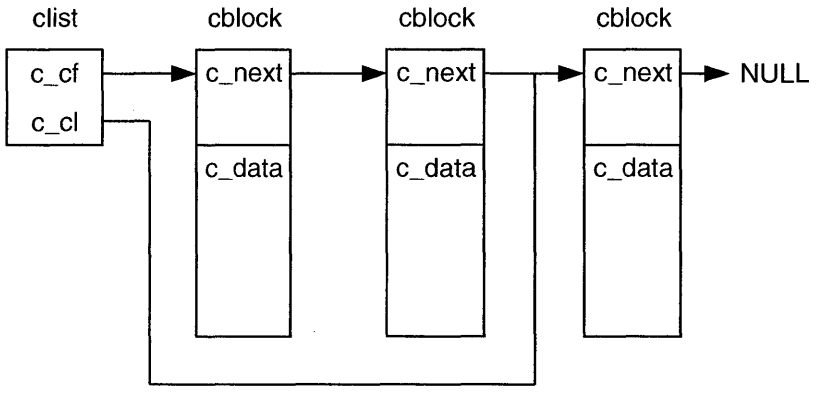


Figure D-4 A character list formed from a chain of character blocks

Examining the values of kernel tunable parameters

Use the **var** command of **crash**(ADM) to view the values of the major tunable kernel parameters:

```
> var
v_buf:  30
v_call:  60
v_clist:  12
v_maxup:  20
v_hbuf:  8
v_hmask:  7
v_pbuf:  2
v_nofiles:  60
v_inode:  75            ve_inode:  75
v_s5inode:  75
v_file:  75             ve_file:  75
v_mount:  4             ve_mount:  4
v_proc:  75             ve_proc:  9
...
```

This report shows the current values of the tunable parameters in the **var** structure, defined in the *<sys/var.h>* header file. The names of the values displayed here correspond to those of the tunable parameter; for example, `v_proc` contains the value of **NPROC** and `v_file` contains the value of **NFILE**. Comments in the header file indicate the tunable for each member and can be used as a reference.

See Appendix B, "Configuring kernel parameters" in the *Performance Guide* for more information about the kernel parameters and how you can change them.

Monitoring memory allocation

Use the **od** command of **crash** to get information about memory allocation on the system. The following commands print the values of the kernel variables that store information about memory allocation; the **-d** option is used to request that decimal values are displayed:

od -d availrmem
Number of 4KB pages of core memory available to user-level processes. This is equal to the total physical memory (**physmem**) excluding memory allocated to the kernel. It does not including swap space.

od -d availsmem
Number of 4KB pages of swappable memory available for use. For diskless clients, **availsmem** and **availrmem** have the same value; on other configurations, **availsmem** represents the unused portions of **availrmem** and **nswap**.

od -d freemem
Number of unused 4KB pages of core memory.

od -d nswap
Size of swap space, as a number of physical 512-byte disk blocks; divide this number by 8 to convert it to its equivalent in units of 4KB pages.

od -d physmem
Total number of 4KB physical memory pages configured on the system. This figure does not change. For example, a 32MB system shows 8096 for **physmem**. Some machines remap part of physical memory so that the kernel cannot locate it; on these machines, **physmem** will have a value lower than the actual physical memory configured on the system. The gap between 640KB and 1MB is an example of this.

Examining use of STREAMS resources

You can use the **strstat** command of **crash**(ADM) to study how STREAMS resources are being used on the system. The output of this command is similar to that produced using the command **netstat -m** as described in the *Performance Guide.*

The following is an example of output from **strstat**:

```
> strstat
ITEM                    CONFIG    ALLOC    FREE      TOTAL     MAX    FAIL   BUFCALL
streams                    160       79      81       2515      89       0   -
queues                     452      374      78       5074     416       0   -
message headers            100       39      61     182337      87       0   -
buffer headers             314      194     120      46826     201       0   -
data block size   64        64        1      63     120940      49       0   -
data block size  128        32        0      32       1538       5       0   -
data block size  256        32        3      29       2995      19       0   -
data block size  512        16        8       8       5875      15       0   -
data block size   1Kb        8        0       8       2846       5       0   -
...
data block size 512Kb        0        0       0          0       0       0   -

Count of scheduled queues:   0
Number of unallocated pages: 466
Buffer splitting threshold: 80
Size of the interrupt pool: 20
Streams daemon (strd) flags:
```

The CONFIG column shows the total number of each resource that was configured for use, and MAX is the maximum number that have been allocated at any one time.

ALLOC is the number that are currently allocated, and TOTAL is the total number that have been used since the system was booted.

FAIL is the number of times that a resource was unavailable since the system was booted. Since data blocks are dynamically allocated from the memory reserved for use by STREAMS, you should not expect to see any value other than 0 in the this column.

See the *Performance Guide* for more information about examining and tuning STREAMS resources.

Translating virtual addresses to physical addresses

Virtual addresses are used by the operating system to access kernel and user memory. The CPU manages translation of virtual to physical addresses using its Memory Management Unit (MMU). A virtual address is specified as a offset from the start of a memory segment; these segments are used by the kernel and user processes to hold their text, stack, data, and other regions. See Figure D-5 (page 451).

The upper 13 bits (3 through 15) of a segment selector holds the entry in the Global Descriptor Table (GDT) used by the kernel, or a process's Local Descriptor Table (LDT). The lower three bits indicate whether the descriptor

table belongs to the kernel or a user process and its privilege level. A segment's descriptor table entry stores (among other things) its base linear address; this is added to the offset to produce a *linear address*. (This intermediate linear addressing enables non-contiguous physical pages to appear contiguous within a segment.)

The upper 10 bits of a linear address index an entry in the page directory table; the base of this table is pointed to by control register 3 (CR3), also known as the page directory base register (PDBR). The contents of this table can be examined using the **sdt** command of **crash**(ADM). The entry in the page directory table points to the base of a page table.

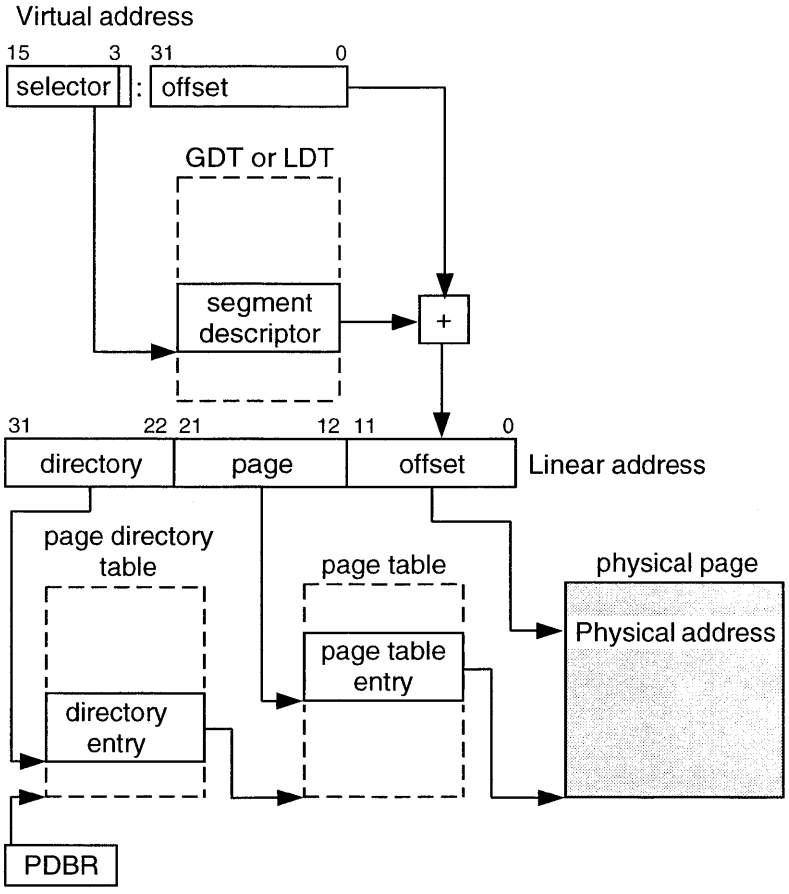


Figure D-5 How a virtual address is translated to a physical address

Bits 12 to 21 of a linear address index an entry in the page table pointed to indirectly by the upper 10 bits. The contents of the page tables can be examined using the **pdt** command of **crash**. The entry in the page table points to the

base address of physical page in memory; the lowest 12 bits of a linear address act as an offset to the location of the physical address within the page.

NOTE You cannot use **crash** to examine a process's page table and pages if they have been swapped out.

Symbols, numbers

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

30 May 1997
AU20021P000