SCO® UNIX®
Operating System

System Administrator's Guide

SCO® UNIX®
Operating System

System Administrator's Guide

© 1983-1993 The Santa Cruz Operation, Inc.
© 1980-1993 Microsoft Corporation.
© 1989-1993 UNIX System Laboratories, Inc.
All Rights Reserved.

No part of this publication may be reproduced, transmitted, stored in a retrieval system, nor translated into any human or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual, or otherwise, without the prior written permission of the copyright owner, The Santa Cruz Operation, Inc., 400 Encinal, Santa Cruz, California, 95060, U.S.A. Copyright infringement is a serious matter under the United States and foreign Copyright Laws.

The copyrighted software that accompanies this manual is licensed to the End User only for use in strict accordance with the End User License Agreement, which should be read carefully before commencing use of the software. Information in this document is subject to change without notice and does not represent a commitment on the part of The Santa Cruz Operation, Inc.

SCO UNIX Software is commercial computer software and, together with any related documentation, is subject to the restrictions on U.S. Government use as set forth below.

If this procurement is for a DOD agency, the following DFAR Restricted Rights Legend applies:

RESTRICTED RIGHTS LEGEND: Use, duplication or disclosure by the Government is subject to restrictions as set forth in subpararaph (c)(1)(ii) of rights in Technical Data and Computer Software Clause at DFARS 252.227-7013. Contractor/Manufacturer is The Santa Cruz Operation, Inc., 400 Encinal Street, Santa Cruz, CA 95060.

If this procurement is for a civilian government agency, the following FAR Restricted Rights Legend applies:

RESTRICTED RIGHTS LEGEND: This computer software is submitted with restricted rights under Government Contract No. _____ (and Subcontract No. _____, if appropriate). It may not be used, reproduced, or disclosed by the Government except as provided in Paragraph (g)(3)(i) of FAR Clause 52.227-14 or as otherwise expressly stated in the contract. Contractor/Manufacturer is The Santa Cruz Operation, Inc., 400 Encinal Street, Santa Cruz, CA 95060.

SCO, Open Desktop, The Santa Cruz Operation, the Open Desktop logo, and the SCO logo are registered trademarks of The Santa Cruz Operation, Inc. in the USA and other countries.

All other brand and product names are or may be trademarks of, and are used to identify products or services of, their respective owners.

Date: 1 February 1993
Document Version: 3.2v4.2B

Introduction xxv

Audience xxv
How to use this book xxvi
Performance and Troubleshooting xxvi
Documentation conventions xxvii

Chapter 1
Understanding system administration 1

The system administrator and administrative roles 1
Summary of administrator's tasks 3
Making administration easier with the sysadmsh 6
Locating manual pages 7
Using online manual pages 7
The super user account 8
The keyboard 9
Running programs simultaneously with MultiScreen 10
System security 12
Educating users 12
What every user should know 12
Site planning considerations 14
Extensions to the UNIX operating system 15

Chapter 2
Using the system administration shell 17

Starting sysadmsh 18
How the screen is organized 19
Selecting menu items 20
Using forms 22
Using radio buttons 28
Using scan windows 29
Getting help 31
Changing the current directory within sysadmsh 33
The function keys 33
Using shell escapes to access the UNIX system command line 34
sysadmsh environment variables 35

Customizing sysadmsh menus 35
sysadmsh menu options 35

Chapter 3
Starting and stopping the system 43

Starting the system 43
Loading the operating system 43
Cleaning filesystems 44
Choosing the mode of system operation 45
Setting the time and date 46
Checking the security databases 47
Logging in as the super user 51
Stopping the system 52
Using the wall command 52
Using the shutdown command 53
Using the haltsys command 53
Understanding the boot display information 54
Changing the boot process 55
Booting an old kernel 55
Changing the default bootstring (DEFBOOTSR) 55
Booting automatically 56

Chapter 4
Administering user accounts 57

How account information is stored 58
Configuring database recovery and precedence 59
Account management 60
Adding a user 60
Removing or retiring a user account 67
Unretiring a user account 67
Locking or unlocking a user account 68
Locking or unlocking a terminal 69
Changing a user's login group 69
Changing a home directory 70
Changing a user password or password parameters 71
Changing user authorizations 74
Changing user audit parameters 75

Adding or changing groups 76
Allowing users to execute super user commands 77
Accessing other accounts with su(C) 78
Moving user accounts to another filesystem or directory 78
Migrating user accounts to non-SCO-based UNIX systems 79
Migrating user accounts to SCO-based UNIX systems 80
Migrating user accounts from SCO XENIX or non-SCO-based UNIX systems 80
Default account configuration 81
Changing the security defaults 83
Changing system account parameters dynamically 85
Changing default login restrictions 85
Changing default password restrictions 87
Changing default authorizations 90
Adding login shells and configuration files 94

Chapter 5
Managing filesystems 95

What is a filesystem? 95
Adding new filesystem(s) 96
Mounting and unmounting a filesystem 97
Permitting users to mount filesystems 98
Filesystem types 99
Converting AFS filesystems to EAFS 99
Converting UNIX filesystems to AFS 99
Configurable filesystem features 100
Setting directory SGID bit 100
Setting filename truncation 100
Using links 101
Symbolic links 101
Maintaining free space in filesystems 102
Strategies for maintaining free space 102
Displaying free space 103
Sending a system-wide message 103
Displaying disk usage 104
Displaying blocks by owner 104
Mailing a message to a user 105
Locating files 105
Checking and clearing log files 108

Maintaining efficient filesystem organization 111
Disk fragmentation 111
Large directories 111
Adding disk space and filesystems 113
Filesystem integrity 114
How UNIX systems maintain files 115
How UNIX systems maintain filesystems 115
Causes of filesystem corruption 116
Rules for checking filesystems 117
Repairing filesystems with fsck 117
Summary of fsck phases 118
Automatic filesystem check 119

Chapter 6

Using printers 121

The printer spooling system 121
Printer configuration options 122
Interface program 123
Printer type 125
Content types 125
Connection method 127
Character sets or print wheels 128
Fault alerting 131
Fault recovery 133
Restricting user access to a printer 134
Default printing attributes 135
Print service command summary 136
Adding a local printer 138
Starting and stopping the print service 142
Manually stopping the print service 142
Manually starting the print service 142
Canceling a print request 143
Enabling and disabling printers 143
Adding a printer to a class 144
Setting the system default destination 145
Mounting a form or print wheel 145
Removing a printer or class 147

Managing the printing load 147
Rejecting requests for a printer or class 148
Accepting requests for a printer or class 148
Moving requests to another printer 149
lpmove, accept and reject examples 149
Managing queue priorities 150
Setting priority limits 151
Setting a default priority 152
Examining the priority limits and defaults 152
Moving a request around in the queue 152
Examining a printer configuration 154
Using forms and filters 155
Using the information in the request log 156
Customizing the print service 157
Adjusting the printer port characteristics 159
Adjusting the terminfo database 161
How to write an interface program 163
Configuring a network printer 170
Configuring a dialup printer 171
Using a printer without the spooler 172
Creating an init device file 172

Chapter 7

Using floppy disks and tape drives 175

Using tape drives 175
Tape device files 175
Changing the default tape drive 176
Setting the default tar(C) device 177
Archiving files on tape 178
Tape drive maintenance 179
Tape driver error correction code (ECC) support 180
Using floppy disks 181
Formatting floppy disks 181
Copying floppy disks 182
Using floppies for file storage 183
Making filesystems on floppy disks 184
Mounting a floppy filesystem 186

Chapter 8
Using MS-DOS and other DOS operating systems 187

OS/2 coexistence 188
Partitioning the hard disk using fdisk 188
Displaying current partition table 189
Switching operating systems 190
Installing a UNIX system partition on a DOS system 192
Using a UNIX system and DOS with two hard disks 193
Removing an operating system from the hard disk 194
DOS accessing utilities 194
File and directory arguments 195
User configurable default file 195
Mounting DOS filesystems on a UNIX system 196
Configuring support for mounted DOS filesystems 196
How DOS filesystems are accessed 197
Using the mount command 197
Repairing and checking DOS filesystems 198
Who can access the mounted DOS filesystem 198
Appearance of DOS files 199
Restrictions 199
UNIX systems and DOS on nonstandard disks 200

Chapter 9
Maintaining system security 201

What is a trusted system? 202
Trusted system concepts 202
Running a trusted system 206
Assigning administrative roles 206
Administering subsystems with sysadmsh 207
Assigning kernel authorizations 208
Controlling system access 209
Logging out idle users 211
Using auditing on your system 212
Protecting the data on your system 212
SUID/SGID and sticky bit clearing on writes 212
The sticky bit and directories 213

Using data encryption 215
Importing data 215
Terminal login management 217
Examining a terminal entry 218
Redefining login attempt limit 219
Locking or unlocking a terminal 219
Setting up device equivalencies database 219
Activity report generation 220
Reporting password status 221
Reporting terminal activity 223
Reporting login activity 224
Detecting system tampering 225
Stolen passwords 226
Abuse of system privileges 227
Unsupervised access to the computer itself 227
Dealing with filesystem and database corruption 227
The authentication database files 228
Checking the system after a crash 229
Using the override terminal 230
Filesystem checking: fsck(ADM) 230
Automatic database checking and recovery: tcbck(ADM) 230
Database consistency checking: authck(ADM) 231
System file integrity checking: integrity(ADM) 231
System file permission repair: fixmog(ADM) 232
Daemon operations on a trusted system 233
LUID enforcement 233
stopio(S) on devices 234
Sticky directories 234
Disabling C2 features 235

Chapter 10
Using the audit subsystem ***237***

Audit subsystem components 238
Audit methodology 242
Guidelines for effective system auditing 246
Performance goals 248
Reliability goals 248

Security goals 249
Administrative concerns 250
Auditing as a debugging tool 251
Data collection 251
Choosing audit events 252
Auditing individual users and groups 253
Displaying current audit statistics 255
Enabling and disabling auditing 255
Maintaining audit trail continuity 256
Adjusting audit performance parameters 257
File and directory management 260
Listing audit sessions 261
Backing up audit files 262
Restoring audit files 263
Removing audit files 263
Monitoring disk space consumption 263
Maintaining collection directories 263
Report generation 266
Creating or modifying a report template 267
Viewing a report template 274
Listing report templates 275
Removing report templates 276
Generating an audit report 276
Understanding audit reports 278
Auditing capabilities extended to users 286
Glossary of audit terms 286

Chapter 11
Administering serial terminals *289*

Changing default terminal line characteristics 289
The gettydefs file 290
Changing the gettydefs file 291
Checking the terminal settings 292
Changing serial line operation 293
Setting the terminal type 294
Setting the terminal type automatically 296
Removing a terminal 296
Setting up a serial console 297

Using serial multiscreens with mscreen 298
Adding pseudo-ttys 299
Troubleshooting 299
Advanced uses 300
Using scancode-compatible terminals 302
Using function keys in scancode mode 302
Correcting a hung scancode-compatible terminal 303

Chapter 12
Using modems ***305***

Choosing a serial port 305
Configuring your modem 306
Smartmodem 1200 or compatible 306
Smartmodem 2400, V-series 9600 or compatible 307
Telebit Trailblazer 308
General modem configuration guidelines 308
Installing a modem 308
Dialing into your computer 314
Adding passwords for dial-in lines 314
Dialing out from your computer 315
Using dialer programs 315

Chapter 13
Building a remote network with UUCP ***323***

What is UUCP? 323
How to use this chapter 324
What you need 325
UUCP commands 325
User programs 325
UUCP directories 326
UUCP background programs 327
How UUCP works 328
A sample UUCP transaction 328
Configuring UUCP on your system 330
Connecting a UUCP modem 330
Editing the UUCP control files 331

Creating login accounts for sites dialing-in 332
UUCP anonymous login accounts 333
Testing the UUCP connection 333
Detailed descriptions of UUCP configuration files 334
Adding entries for remote sites to the Systems file 334
Setting up polling 340
Limiting access with the Permissions file 341
Adding dial-out entries to the Devices file 348
Special UUCP configuration options 352
Defining a communications protocol 352
Adding dialers to the Dialers file 353
Using Dialcodes to create a portable Systems file 353
Creating alternate control files with Sysfiles 353
Changing uucico packet parameters 354
Preventing unknown sites from logging in 354
Configuring UUCP over TCP/IP 354
Connecting two local systems using a direct wire 360
Complete UUCP examples 362
Example 1: system gomer 362
Example 2: system dingbat 364
Sample commands 366
Administering your UUCP system 366
UUCP maintenance shell scripts 366
Generating log reports on usage: uulog 367
The UUCP spool directory 367
UUCP error messages 369
ASSERT error messages 370
UUCP STATUS error messages 372

Chapter 14
Setting up electronic mail ***375***

Chapter overview 376
How MMDF works 376
Outgoing mail 377
Incoming mail 380
The configuration utility 381
Configuring communications channels 382
Configuration utility limitations 382

MMDF configuration checklist 383
Host name 385
Domain name 385
Fully qualified host name 386
Hide your host name? 388
Redirect mail for root? 389
Redirect mail for mmdf? 389
Postmaster address 389
Setting up MMDF for UUCP 390
Configuring MMDF to use a name server 390
Configuring smart hosts 391
Running the configuration utility 392
Running the configuration utility with TCP/IP installed 399
Preparing MMDF for use 401
Other common configuration tasks 402
Rebuilding the MMDF hashed database 403
MMDF configuration files 404
The mmdftailor file 404
Alias files 412
Domain files 415
Channel files 418
How MMDF routes mail 420
Searching MMDF domain tables 421
Delivery channel programs 422
Modifying MMDF configuration 424
Guidelines for manual configuration 425
Changing the postmaster alias 425
Maintaining user aliases 425
Changing the host name 426
Hiding your host name 427
Changing your smart host 427
Changing error logging levels 428
Converting from XENIX 428
Modifying UUCP host configuration 431
Adding or removing an SMTP host 432
Running multiple deliver daemons 432
Rebuilding the MMDF hashed database 434
Testing MMDF configuration 434
Checking for MMDF problems 434
Testing mail addresses 435

Maintaining the MMDF system 435
Checking the status of mail queues 435
Removing old mail from the queues 436
Monitoring the size of log files 436
Advanced MMDF configuration tasks 436
Changing the location of system mailboxes 437
Specifying the MMDF "signature" 437
Configuring MMDF authorization 437
Adding a new alias file 446
Adding a new domain file 448
Adding a third-party channel program 448
Changing MMDF parameters 449
Getting more information 451

Chapter 15
Backing up filesystems **453**

Strategies for backups using sysadmsh 454
Using the backup authorization 454
Floppy drive backups and large systems 455
Summary of utilities accessed 455
Preparations for scheduled backups 456
Creating a backup schedule 456
Labeling your backups 461
Keeping a log book 462
Rotating backup media 463
Archiving backup media 463
Performing a scheduled backup 463
Using formatted media 463
Starting the backup 464
Performing an unscheduled backup 465
Verifying a backup 468
Getting a backup listing 469
Restoring individual files or directories from backups 471
Restoring an entire filesystem 473
A detailed explanation of backup levels 474
Principles of incremental backup levels 474
How the default and alternate schedules work 475
How backups restore a filesystem 476

Unattended backups 477
The cbackup shell script 477
Example crontab entries 478

Chapter 16
Troubleshooting overview *481*

Avoiding system problems 481
Preparing to recover from problems 482
Kernel error messages 483
Recovering from a system crash 485
Recovering from a system panic 486
Recovering after a power failure 487
Verifying the security system after a crash 487

Chapter 17
Performance management *489*

General performance considerations 489
Understanding your performance goals 490
Excessive paging 491
Kernel tables are too small 493
Tuning disk I/O parameters 494
Optimizing serial I/O 496
Increasing the value of TTHOG and NCLIST 497
Changing the sio_fifoctl array 498
Relinking the kernel 501

Chapter 18
Tools reference *503*

configure(ADM) — reallocate kernel resources 504
Using the configure command line 505
Overriding configure limit warnings 505

crash(ADM) — view memory 505
Initializing crash 505
crash strstat (statistics about STREAMS buffers) 506
var (values of tunable parameters) 508
fsck(ADM) — check and repair filesystems 508
idtune(ADM) — reallocate kernel resources 509
integrity(ADM) — check system file permissions 509
ps(C) — check process activity 510
sar(ADM) — system activity reporter 511
How sar works 512
Running sar 512
sar -a — file access operations 513
sar -b — buffer activity 513
sar -c — system call activity 514
sar -d — block device activity 515
sar -m — interprocess communication 516
sar -n — name cache activity 517
sar -p — paging activity 517
sar -q — process throughput 518
sar -r — page and swap usage 519
sar -u — CPU utilization 519
sar -v — system tables 520
sar -w — swapping activity 521
sar -y — terminal I/O 522
swap(ADM) — check and add swap space 523
timex(ADM) — examine system activity per command 523
XENIX system tools 524

Chapter 19
Kernel parameter reference ***525***

Disks and buffers 528
Character buffers 531
Files, inodes, and filesystems 532
Processes, memory management, and swapping 535
Clock 538
MultiScreens 538
Message queues 539
Semaphores 540
Shared data 541

System name 542
STREAMS data 542
Event queues and devices 545
Hardware dependent parameters 546
Security 547
Asynchronous I/O 548
Boot load extension parameters 548
Multiprocessing parameters 549

Chapter 20
Troubleshooting reference *551*

Boot troubleshooting 551
Restoring missing or corrupted system files 552
/boot not found 552
unix not found 553
Cannot load floating point emulator 554
System hangs at login prompt 555
System cannot enter multiuser mode 556
execlp of /bin/sulogin failed 557
/etc/bcheckrc not found 557
System hangs at boot time 558
Console troubleshooting 558
Console keyboard locks up 559
Preventing console keyboard lockup 560
Wrong console keyboard type 560
Cannot log into console 561
cron, at, and batch troubleshooting 561
cron daemon is not running 561
at command fails: cannot change mode of job 562
Disk problems 562
Mapping a bad track 562
Disk load balancing 563
Filesystem troubleshooting 564
Out of inodes on filesystem 564
Out of space on filesystem 565
Checking free space on filesystems 567
Reducing disk fragmentation 567
Restoring a corrupted root filesystem 568
Repairing a filesystem when fsck stops at size check 571

Filesystem mount failed 574
Unable to remove files 575
Login problems 575
Cannot log in after entering multiuser mode 575
Users cannot log into the system 576
Login incorrect 576
useshell: file access control database inconsistency 576
Unlocking a locked user account 576
Replacing a forgotten user password 576
Mail troubleshooting 577
Failed mail error 577
Mail does not work, no returned mail 578
Mail command hangs 578
Modem problems 579
Modem — errors when dialing out 579
No OK message (Hayes-compatible modems) 579
Modem dials, but does not connect 580
Connect failed: NO DEVICES AVAILABLE 580
Modem answers, but terminal displays garbage 581
DEVICE LOCKED 581
Modem does not hang up 581
Double echo 582
Modem — problems dialing in 582
Modem does not answer the phone 582
Modem answers, but hangs up 582
Garbage or loose cable 583
Modem answers, but no login prompt 583
Screen displays a series of login prompts 584
System displays meaningless characters 584
Mouse (serial) troubleshooting 585
usemouse: no mouse available(5) 585
usemouse: can't open slave (errno 13) 585
No more processes error message 586
Runaway processes 586
Print system troubleshooting 587
lpsched print scheduler is not running 587
Printer does not print 587
Cannot redirect output to printer 589
Port does not respond 590
Printer output is illegible 590
Printer output spacing is wrong 591

Parallel printer is slow ... 592
Printer reports UUCP errors ... 594
Runaway processes, stopping ... 594
Security system troubleshooting ... 595
Using the override terminal ... 595
Account is disabled -- see Account Administrator ... 596
Account is disabled but console login is allowed Terminal is disabled but root login is allowed ... 596
Audit: filesystem is getting full ... 596
Authentication database contains an inconsistency ... 597
Can't rewrite terminal control entry for tty Authentication error; see Account Administrator ... 597
Cannot obtain database information on this terminal ... 598
Login incorrect ... 598
login: resource Authorization name file could not be allocated due to: cannot open; ... 598
Terminal is disabled -- see Account Administrator ... 598
useshell: File Control database inconsistency ... 598
useshell: Script path /usr/lib/mkuser/mkuser.init may be compromised 598
You do not have authorization to run 599
Tape drive problems ... 599
Bad octal digit ... 599
Tape not recognized at bootup ... 599
Tape commands hang ... 600
Cannot open /dev/rct0 error message ... 601
Terminal troubleshooting ... 601
Restoring a non-functional terminal ... 602
Fixing a hung terminal ... 602
Fixing a scrambled terminal display ... 603
Unlocking a locked terminal ... 604
Restoring a non-echoing terminal ... 604
UUCP troubleshooting ... 604
Check for faulty ACU/Modem ... 605
Errors when testing the connection with cu ... 606
UUCP failed messages ... 607
Checking the status of a uucp request ... 608
Debugging UUCP communications ... 609
Alarms in UUCP audit output, data is not transferring ... 610
UUCP log and status file messages ... 610
Checking UUCP files and permissions settings ... 613
Verifying that sitename is unique ... 613

UUCP truncates system names to seven characters 613
UUCP is abnormally slow 614
uucp works, but uux does not 615
UUCP troubleshooting utilities 615

Appendix A
Customizing system startup 619

Changing the /etc/inittab file 620
Changing scripts in /etc/rc2.d 623
Starting daemons on a trusted system 626
Daemons that must run without an LUID 626
Modifying .profile and .login files 626
Changing the /etc/motd file 627

Appendix B
Authorizing the use of job scheduling commands 629

Changing default authorization for job scheduling 629
Changing cron defaults 629
Changing the at or batch defaults 630
Allowing or denying individuals the use of cron 630
Viewing user cron permissions 631
Allowing or denying individuals the use of at or batch 631
Viewing user at or batch permissions 631
Using environment files for the at or batch commands 632
Example of environment file usage 632

Appendix C
Using the system console and color displays 633

Console keyboard type selection 634
Switching keyboard modes manually 634
Changing modes permanently 634
Using the console screen protection feature 635

Using MultiScreen 635
Reducing the number of multiscreens 636
Multiscreens and multiple video adapters 637
Changing video fonts 637
Controlling color displays with setcolor 638
Changing the foreground and background colors 638
Changing reverse video colors 639
Changing the screen border color 639
Sounding the keyboard bell 639
Resetting the screen 639

Appendix D
UNIX directories and special device files **641**

UNIX directories 641
The root directory 641
The /bin directory 642
The /dev directory 642
The /etc directory 643
The /lib directory 644
The /mnt directory 644
The /usr directory 645
The /tcb directory 645
The /tmp directory 645
Log files 646
Special device files 647
Special filenames 647
Block sizes 647
Gap and block numbers 648
Serial devices 648

Index **649**

Introduction

The UNIX system is designed to accommodate a wide variety of needs, including the use of peripheral devices such as terminals, printers, and tape drives. Your UNIX system requires careful control of its operation and a regular schedule of maintenance. This guide explains how to run and maintain the operating system on your computer, ensuring maximum performance with a minimum number of problems.

Audience

This guide is written for system administrators. The tasks presented range from simple ones requiring very little knowledge about UNIX systems, to complex tasks requiring extensive knowledge about the operating system and your computer. Each chapter explains the tools and knowledge you need to complete the tasks described in that chapter.

Performance management and troubleshooting require a considerable knowledge of the internal workings of your system. While we have attempted to provide the information needed for these tasks, users who are attempting to enhance the performance of their systems and troubleshoot error conditions may need to consult other sources to get a fuller understanding of how the system works. The information in this book will help you get the highest performance possible from your machine and recover from problems that result from a power failure, system failure, or human error.

NOTE The *Hardware Configuration Guide* explains how to connect and configure devices such as serial cards, hard disks, terminals, and tape drives. This guide is concerned with the use of such devices after they are installed. For example, the "Using floppy disks and tape drives" chapter of this guide discusses the use of magnetic storage media, and how to use a floppy or tape drive once it is installed.

How to use this book

The *System Administrator's Guide* is divided into three sections:

- System configuration and maintenance (chapters 1 through 15)
- Performance and troubleshooting (chapters 16 through 20)
- Appendices (A through D)

The guide is designed to be used as follows:

- Use the "Understanding system administration" chapter to learn how to administer your system.
- Each area of system operation has a separate chapter in "System configuration and maintenance." For example, the "Using printers" chapter covers setting up, troubleshooting, adding, removing, customizing, error recovery and more.
- Use the index when you are seeking information about a specific topic; the index includes entries for all error messages that are discussed in the book and other detailed entries to enable you to find the information you need quickly.
- Read the "Performance and Troubleshooting" chapters carefully *before* you have problems; understanding this material may help you avoid serious problems later.

Performance and Troubleshooting

The Performance and Troubleshooting section (chapters 16 through 20) contains important information about solving system problems:

Troubleshooting overview
: provides general information about avoiding system problems and being prepared to handle problems quickly. It also gives detailed instructions for recovering from a system panic or power outage.

Performance management
: discusses general steps you can take to improve system performance and discusses the general approach you should take to resolve performance problems. Many different situations can cause a degradation in performance; to solve the problem effectively, you must first analyze it to determine what the underlying cause is. If you determine that your system needs to be reconfigured to allocate system resources differently, this chapter also includes information about the adjustments that are most commonly required.

Tools reference
: alphabetically arranged material about system tools that help you analyze and correct system problems.

Kernel parameter reference
: comprehensive listing of tunable parameters whose value can be modified to reallocate kernel resources appropriately. These parameters are discussed in the groupings and order in which they appear when you run the **configure**(ADM) program.

Troubleshooting reference
: reference sections about various topics that can cause system problems. These sections are arranged alphabetically.

Documentation conventions

The following documentation conventions are used in this guide.

boldface Commands are shown in **boldface**. For example:

... the **fsck** command examines the various structures on the disk and ...

UNIX system utilities or library routines are also shown in **boldface**. For example:

... the installation program runs the **fdisk**(ADM) utility ...

Notice in this example that the location of the corresponding manual page is also given. A full list of manual page locations is given in Table 1-2.

Literal user input is also shown in **boldface**. For example:

... to display the file itself, enter:

more /etc/termcap

and press ⟨Return⟩ ...

italics

Directories and filenames are shown in *italics*. For example:

... the set of administrative print service commands is contained in the */usr/lib* directory ...

Emphasized words or phrases are also shown in *italics*. For example:

... the constant creation and removal of files creates a situation called *disk fragmentation* ...

References to book titles are also shown in *italics*, but with initial capitals. For example:

... for information relating to system use, refer to the *User's Guide* ...

bold italics

Placeholders are shown in ***bold italics***. A placeholder is a word which you must replace with an appropriate filename, number, or option. For example:

... do you want to use the ***mouse_type*** on any other terminals ...

In this example, ***mouse_type*** would be replaced by a brand or type of mouse.

`courier`

Screen displays and other output from the computer are shown in `courier`. For example:

```
      The UNIX kernel has been rebuilt.
Do you want this kernel to boot by default? (y/n)
```

`reverse video`

When you are using **sysadmsh**, the context indicator, status line, and cursor position are shown in `reverse video`. See Chapter 2 of this manual for examples.

" "

Data values and field names are shown in "quotation" marks. For example:

... where *x* is "0" for a display adapter or "1" for a serial port ...

Quotation marks are also used for normal words used in a way particular to computing. For example:

... the former is called the "master" tty and the latter is called the "slave" tty ...

Document chapter names are also shown in quotations. For example:

... consult the "Using floppy disks and tape drives" chapter of this guide ...

SMALL CAPITALS

Acronyms are shown in SMALL CAPITALS. For example:

... the name UUCP is an acronym for UNIX to UNIX Copy ...

SMALL BOLD CAPITALS

System parameters (definable system values, for example, the number of disk drives attached to the system), named constants (programming names with fixed values) and environment variables (definable system information, for example, what type of terminal is being used) are shown in **SMALL BOLD CAPITALS**. For example:

... the **mkdev ptty** updates the **NSPTTYS** kernel parameter ...

... the preferable method for setting your terminal type is to assign the type to the **TERM** variable ...

⟨ ⟩

Names of keys are shown in ⟨angle brackets⟩. For example:

... press the ⟨Esc⟩ key to exit the current mode ...

Δ

Where **sysadmsh** menu selections are given as alternatives to command line entries they are indicated by Δ. For example:

... to copy all the files in the directory */u/bogart* to the cartridge drive */dev/rct0*, enter:

tar cvf /dev/rct0 /u/bogart

Δ **sysadmsh** users select: Media ⇨ Archive ...

Configuration and Maintenance

Chapter 1

Understanding system administration

This guide is designed to cover tasks that the system administrator would need to perform to ensure the smooth and efficient running of the system.

This guide explains how to expand your system with remote communications over phone lines; UUCP can be set up to communicate with UNIX sites all over the world. (See Chapter 13 (page 323) for a complete explanation of network facilities available.)

An important part of system operation is the protection of data on the system. Security is discussed in great detail in Chapter 9 (page 201). The system includes flexible mechanisms designed to protect your data.

Pay special attention to Chapter 20 (page 551). This chapter is an excellent resource to help you keep your system running smoothly. Refer to this chapter whenever you run into difficulties.

The system administrator and administrative roles

Every UNIX system should have at least one person in charge of system maintenance and operation. In this guide, such a person is called a system administrator. It is the responsibility of system administrators to ensure the smooth operation of the system and to perform a wide variety of tasks that require special privileges.

You can choose to have a single system administrator or divide the tasks among several persons, each charged with a different area of operation. You can even assign roles that are strictly limited to one aspect of the system.

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

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

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

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

- Make certain the integrity of the system is not compromised through use of security mechanisms.
- Make certain that adequate backups (regular copies of files on the system) are made and stored for future use.
- Handle problems related to use of limited computer resources (disk space, number of processes, and so on).
- Alleviate system communication (network) stoppages due to failed connections.
- Apply operating system updates and maintenance fixes.
- Provide general support to users.

Summary of administrator's tasks

A system administrator has numerous tasks to perform. They can be divided into groups according to how often they are carried out. The following list of tasks ranges from those that must be performed more often than once a day to those that need be performed less often than once a month. The administrator may have to do some of the tasks in the following list more or less often, depending upon the size and complexity of the system. For more information on each task, see the reference indicated.

Table 1-1 Task list

As needed tasks	For more information:
☐ Record all system modifications and events in log.	Preceding paragraphs in this section
☐ Be on call to restart the system after panics, crashes, power spikes.	"Starting the system" (page 43) and "Recovering from a system crash" (page 485)
☐ Maintain security of hardware, software, data file access.	Chapter 9 (page 201)

Daily tasks	For more information:
☐ Perform scheduled backups (if configured).	"Performing a scheduled backup" (page 463)
☐ Check usage levels.	"ps(C) — check process activity" (page 510)
☐ Check for runaway processes.	"Runaway processes" (page 586)
☐ Check disk space.	"Checking free space on filesystems" (page 567)
☐ Check mail functionality, connections.	"Maintaining the MMDF system" (page 435)
☐ Check printer status.	"Examining a printer configuration" (page 154)
☐ Check auditing output, if activated.	"Report generation" (page 266)

(Continued on next page)

(Continued)

Daily tasks	For more information:
☐ Check communications links, if active.	"Generating log reports on usage: uulog" (page 367)
☐ Check for unattended login sessions.	Man page for **who**(C) and "Activity report generation" (page 220)
☐ Remove *core* and **.out* files.	"Locating core files" (page 108)

Weekly tasks	For more information:
☐ Run **fsck**(ADM) on all filesystems.	"Filesystem integrity" (page 114)
☐ Check printer spooler status report.	Check **lp**(C) account mailbox for messages
☐ Check log files such as */etc/wtmp* and those in */usr/adm* and */usr/spool* and clear, trim, or truncate.	"Checking and clearing log files" (page 108)
☐ Use **sar**(ADM) to generate a report of activity.	"sar(ADM)" (page 511)
☐ Generate detailed report of user disk utilization.	"Displaying blocks by owner" (page 104)
☐ Remove temporary files and *lost+found* files.	"Finding temporary files" (page 106)

Monthly tasks	For more information:
☐ If you are not doing scheduled backups, perform an unscheduled (level 0) backup.	"Performing an unscheduled backup" (page 465)
☐ Archive critical files if changed.	"Archiving files on tape" (page 178) or "Using floppies for file storage" (page 183)
☐ Re-tune system and re-allocate resources, if necessary.	Chapter 17 (page 489)
☐ Perform tape drive maintenance.	"Tape drive maintenance" (page 179)
☐ Change dial-in passwords, if necessary.	"Adding passwords for dial-in lines" (page 314)
☐ Change root password, if necessary.	"Changing a user password or password parameters" (page 71)

Occasional tasks	For more information:
☐ Upgrade OS and Application software, as needed.	Installation notes of the new version
☐ Fix permissions on software.	**fixperm**(ADM)
☐ Re-distribute space in filesystems.	"Checking free space on filesystems" (page 567) or "Maintaining free space in filesystems" (page 102)
☐ Find SUID or SGID files, check owner, size. Find "orphan" files (no real user).	"Locating files" (page 105)
☐ Locate huge (over 64 Mbyte) files and verify their purpose.	"Locating files" (page 105)
☐ Find "orphan" files (no real user).	"Locating files" (page 105)

Making administration easier with the sysadmsh

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

If you are new to UNIX operating systems, we strongly recommend that you become familiar with the concepts and tasks covered in the *Tutorial*. This ensures that you can understand and perform the basic system administrative tasks described here.

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

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

Enter the following command:

custom

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

This means that you can access the functions of the **custom** command by first selecting System at the main **sysadmsh** menu, followed by selecting Software at the next lower level. Selections can be made from the menu in any of the following ways:

- Move through the menu options using the ⟨Space⟩ key and press ⟨Return⟩ on the option you want.
- Move left and right through the options using the arrow keys and press ⟨Return⟩ on the required option.
- Press the first letter of the option required. This is the quickest way. Using the example above, you would simply enter **ss** (without the ⟨Return⟩ key) to reach the **custom** menu.
- Move through the menu options with a mouse, clicking the left button to select an option.

For more instructions on using the **sysadmsh**, refer to the "Using the system administration shell" chapter in this guide.

Locating manual pages

When you use the command line rather than the **sysadmsh** menu interface, you have direct access to utilities and data. Notice the form used for commands in this guide. Each command is printed in bold type, and each has a suffix to help you find more information about it.

The following table lists the locations of the manual pages for the commands with the indicated suffixes. To find information about a command, note the letter or letters that appear in parentheses following the command, then look up the command in the appropriate reference book or guide. For example, the command **lpstat**(C) is defined in the Commands (C) section of the *User's Reference.*

Using online manual pages

If manual pages are installed on your system they may be viewed by typing:

man ***command***

where ***command*** is the command for which you want to see the manual page.

For example, to see the manual page for the **more** command, type:

man more

Some manual pages appear in more than one section (see the following table). To see all occurrences of a particular manual page, type:

man -a ***command***

For example, to see all the manual pages for the **hd** command, type:

man -a hd

The order (by section) in which manual pages are displayed is determined by the */etc/default/man* file. You may edit this file to alter the display order.

To force the system to display a manual page for a particular section, type:

man ***section command***

For example, to see the manual page for the **hd** command for the HW section, type:

man HW hd

Table 1-2 Manual page sections

Command suffix	Book and purpose
ADM	*System Administrator's Reference* — commands reserved for the exclusive use of system administrators
C	*User's Reference* — operating system commands available to all users
CP	*Programmer's Reference* — programming commands used with the development system
DOS	*Programmer's Reference* — DOS routines used with the development system
F	*System Administrator's Reference* — (File Formats) description of system configuration files
FP	*Programmer's Reference* — (File Formats) description of system files and data structures
HW	*System Administrator's Reference* — information about hardware devices and device nodes
K	*Device Driver Writer's Guide* — routines provided in the kernel for writing device drivers
M	*User's Reference* — miscellaneous information used for access to devices, system maintenance, and communications
S	*Programmer's Reference* — system calls and library routines for C and assembly language programming

NOTE The *Programmer's Reference* and *Device Driver Writer's Guide* are only supplied if the Development System is purchased.

The super user account

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

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

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

The keyboard

Many keys and key combinations perform special actions on UNIX systems. These actions have names that may not correspond to the keytop labels on your keyboard. Table 1-3 shows which keys on a typical terminal correspond to special actions on UNIX systems. A list for your particular login device is in **keyboard**(HW). Many of these keys can be modified by the user; see **stty**(C).

Table 1-3 Special keys

UNIX name	Action
⟨Return⟩	terminates a command line and initiates an action. This key is also called the ⟨Enter⟩ key; the keytop may indicate a down-left arrow.
⟨Esc⟩	exits the current mode; for example, exits insert mode when in the editor **vi**. This is also known as the ESCAPE key.
⟨Del⟩	stops the current program, returning to the shell prompt. This key is also known as the INTERRUPT key.
⟨Bksp⟩	deletes the character to the left of the cursor. The keytop may show a large left arrow, as opposed to the small "cursor left" arrow.
⟨Ctrl⟩**d**	signals the end of input from the keyboard; exits the current shell, or logs you out if the current shell is the login shell.
⟨Ctrl⟩**h**	deletes the first character to the left of the cursor. This is also called the ERASE key.
⟨Ctrl⟩**q**	restarts printing after it is stopped with ⟨Ctrl⟩**s**.
⟨Ctrl⟩**s**	stops printing at the standard output device, such as a terminal. This keystroke does not stop the program.
⟨Ctrl⟩**u**	deletes all characters on the current line. This is also called the KILL key.

(Continued on next page)

Table 1-3 Special keys
(Continued)

UNIX name	Action
⟨Ctrl⟩\	quits current command, creates a *core* file. This is also called the QUIT key. (Use of this keystroke is recommended for debugging only; see **core**(FP).)

Running programs simultaneously with MultiScreen

With MultiScreen™, you can run several programs on your console at the same time. You see the display for each program on a different "screen," but you never have to leave your single console. Pressing a simple key combination switches you from one screen to another, and each screen acts independently from the others.

When you log in normally, you see the screen associated with the ⟨Alt⟩⟨F1⟩ key combination. To open a second screen, press and hold the ⟨Alt⟩ key, then press ⟨F2⟩ or another function key on your keyboard. Function keys are generally located across the top or down the far left side of your keyboard.

After you press the key combination, the screen clears and a login prompt appears. Log in again and you can begin work on the second "screen." Press ⟨Alt⟩⟨F1⟩ to switch instantly back to the first screen. Switching between screens in this way can speed up procedures that require working in two different programs. You can run both programs simultaneously, and work on either one at any time. See Figure 1-1.

For example, you can start **sysadmsh** on your first screen, then press ⟨Alt⟩⟨F2⟩ to create a second screen. Log in again and begin editing a log file with **vi**. Use ⟨Alt⟩⟨F1⟩ and ⟨Alt⟩⟨F2⟩ to switch back and forth between screens, taking actions in **sysadmsh** and recording them in the log.

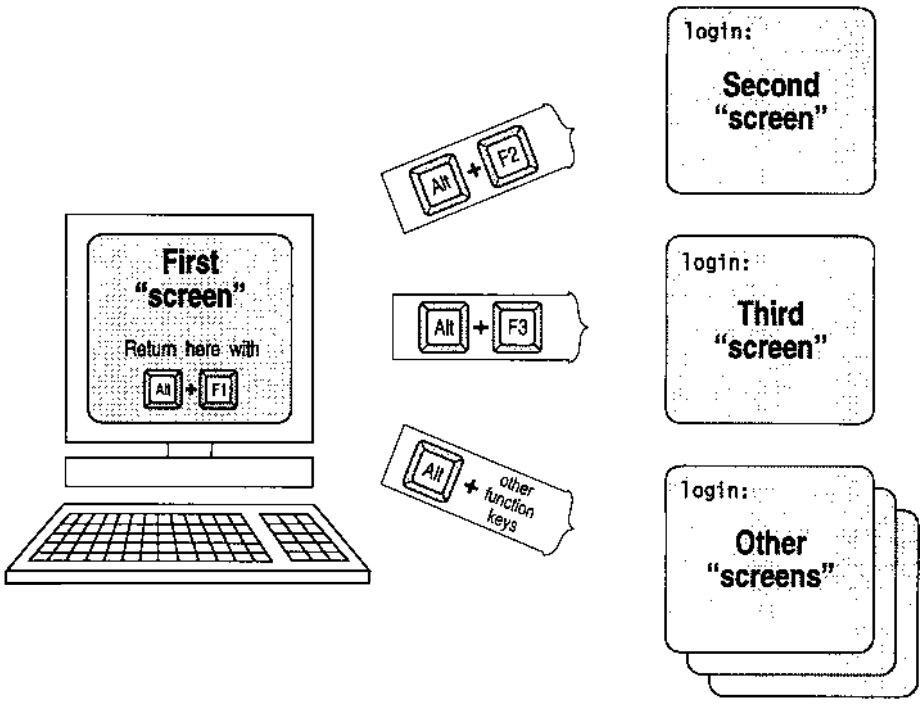


Figure 1-1 MultiScreen example

You can open more than two multiscreens at once. However, make sure to log out of all extra screens before logging out of your first screen. Unattended screens where you remain logged in allow unauthorized access to the system.

If you have several screens open at once, you can rotate through them by pressing the Control and Print Screen key combination, ⟨Ctrl⟩⟨PrtSc⟩. This is helpful if you cannot remember which process is running on a particular screen.

Note that you can configure the system to use ⟨Ctrl⟩⟨Alt⟩***function-key*** combinations in addition to ⟨Alt⟩***function-key*** combinations to change multiscreens. This is especially useful in applications that reserve the ⟨Alt⟩***function-key*** combinations for their own use. This can be configured using the **mapkey**(ADM) utility.

For more information, refer to **multiscreen**(M) and **screen**(HW).

System security

An important consideration is protecting the system and its data from unauthorized access. This system includes security mechanisms not found on other UNIX systems. These mechanisms are designed to meet the C2 class of "trust" as defined by the Trusted Computer System Evaluation Criteria (also known as the *Orange Book*). As the system administrator, you can configure the protection mechanisms to the requirements of your site. You can also set up the powerful auditing features to keep detailed records of logins and system usage. The "Administering user accounts" chapter explains how to add users to the system and how to configure the default security scheme. The "Maintaining system security" chapter covers all aspects of trusted operation, and the "Using the audit subsystem" chapter explains how to use the auditing features.

Educating users

The following list contains items the system administrator can explain to users so they can take advantage of the system's resources without overloading them or causing unnecessary system problems. The more users understand the system and its limits, the less demands are placed on the system administrator.

If a user is completely new to UNIX systems, the system administrator should recommend a training course or at least a careful review of the *Tutorial*. Augment this training with the items listed here.

What every user should know

- System security level — Inform the user whether security is relaxed or not, and if it is not, what files and directories the user can access, and what system actions can be performed. See Chapter 9 (page 201).
- How to log in — Describe, if permitted, how to generate a user password. Emphasize password secrecy, and methods of memorization. Explain the limits on password length and type, the schedule for changing passwords, rules about changing passwords, the number of tries allowed. Tell users how to specify their terminal type, and help them preset the terminal type in their *.login* or *.profile* files if they use the same type of terminal regularly. Point users to the "Getting started" chapter of the *Tutorial*.
- How to manage files and directories — Make sure the user is familiar with basic commands. Refer the user to the *Tutorial* if necessary.

- How to edit files — Make sure the user knows basic **vi** commands, or the basic commands of the default editor. Refer the user to the *Tutorial*, the *User's Guide*, or a training class, if necessary.
- How to invoke programs — Relate the names, locations, and commands necessary to run the most-used programs.
- How to print files — Explain appropriate print commands for output of most-used programs. Tell the user the location of the default print service printer. Demonstrate how to replenish paper and toner, tape, or ribbon cartridge. Refer the user to the printer documentation, if necessary.
- Good mail etiquette — Explain how to read and send mail, how to glean through mail lists, and how to organize messages by appending them to mailbox files. Demonstrate how these files can be read with the **mail -f** command. Explain how to clean out unwanted messages by deleting or saving them to floppy. Emphasize the importance of maintaining free disk space.
- Good filesystem planning — Explain the limits of directory size. For best performance, login and working directories should have less than 64* entries (including the dot (.) and dot dot (..) entries), and data storage directories should have less than 638* entries. Warn users that directories do not get smaller, even if entries are removed. Discourage users from saving mail messages in separate files rather than appending them to existing mailbox files. See "Maintaining efficient filesystem organization" (page 111).
- How to store (archive) files — Show users how to use **tar**(C) to archive unused files or directories to tape or floppy disk. Refer users to the *User's Guide*.
- How to reset a scrambled terminal — Teach users how to escape from most-used programs and how to reset a terminal with **tset**(C) and **stty sane**. Tell users how to turn the terminal on and off if necessary. Make sure users try these procedures before asking a system administrator to disable and reenable the scrambled terminal. See "Fixing a scrambled terminal display" (page 603).
- How to kill a hung process — Instruct users how to use **ps -flu** on a neighbor's terminal to find the hung process on their own terminal, and how to kill the process without causing undue system problems. See "Fixing a hung terminal" (page 602).

* These figures apply to filenames of 14 characters or less. As filename lengths increase, up to a maximum of 255 characters, the number of files that fit on a single disk block decreases, thus reducing the optimum number of files in a directory.

Site planning considerations

Before installing or expanding your system, review the following list of considerations with management and create a physical plan of the system that makes the most efficient use of available resources, and allows users the most direct and complete access to those resources.

If growth is expected, plan for it. Make sure that the resources and the communications lines that connect them are adequate for expected growth.

- Set up the computers in a place where they will not be bumped or moved at any time. If possible, they should be in a room by themselves, with little or no foot traffic. If workstations are used for data storage, at least place them on stable furniture and leave no cables exposed to traffic.
- Keep the computer room cool and give each machine excellent ventilation; keep all machines away from walls and, if possible, provide a separate air conditioner for the computer room, with more-than-adequate cooling capability.
- Install a Halon fire extinguishing system in the computer room rather than sprinklers.
- Store backup media in a separate room from the computers. This room should be fireproof, or should at least have a Halon fire extinguishing system rather than a sprinkler system.
- Ensure that there is adequate and uninterrupted power for the computers. This means more than enough current, and at least surge suppressors, if not a means of guaranteeing uninterrupted power supply. If power fluctuations and failures are common, provide a backup power supply. The computers should also be on an isolated, fully grounded (earthed) circuit.
- If you install a Local Area Network, plan the cabling and location of all machines and peripherals carefully. Seek the assistance of a networking expert to make these plans. Good planning and the use of adequate connecting media and compatible hardware are essential for long-term network performance.
- If you need modem lines for off-site connections, arrange for these with your local telephone company.
- If you plan to connect a printer to a parallel port, locate it close to the machine running it, but keep the machine out of the path of traffic to and from the printer.

- If you connect terminals, printers, or other peripherals to serial ports, consider using phone-line cabling and switching hardware, especially if your system is expected to grow in size and complexity. You can readily adapt phone lines for serial hardware, and telephone connecting and switching technology is mature and flexible.

Extensions to the UNIX operating system

A number of features described in this manual represent extensions to the AT&T System V/386 UNIX Release 3.2 base. These features are added value from the Santa Cruz Operation, Inc., and are summarized in Table 1-4. Features that are part of the AT&T System V UNIX Release 3.2 base, but include added value, are marked with a dagger (†).

Table 1-4 Value-added features

addxusers(ADM)	dbmbuild(ADM)	man(C)	sd(ADM)
ale(ADM)	default(F)	mapkey(ADM)	screen(HW)
ap(ADM)	deliver(ADM)	mcconfig(F)	setcolor(C)
asroot(ADM)	dos(C)	menumerge(ADM)	sg(C)
audit(HW)	dtox(C)	mkdev(ADM)	shutdown(ADM)†
auditd(ADM)	eisa(ADM)	mkfs(ADM)†	su(C)†
authcap(F)	fdisk(C)†	mmdfalias(ADM)	submit(ADM)
authck(ADM)	filesys(F)	mmdftailor(F)	subsystem(M)
authckrc(ADM)	fixmog(ADM)	mnlist(ADM)	sysadmsh(ADM)
autoboot(ADM)†	fixperm(ADM)†	mscreen(M)	tables(F)
badtrk(ADM)	fsave(ADM)	multiscreen(M)	tail(C)
boot(HW)†	fsck(ADM)†	passwd(C)†	tape(C)
cdrom(HW)	fsphoto(ADM)	prwarn(C)	tcbck(ADM)
checkaddr(ADM)	goodpw(ADM)	queue(F)	ttyupd(ADM)
checkmail(C)	hwconfig(C)	rcp(C)	uname(C)†
checkque(ADM)	idleout(M)	rcvalert(C)	unretire(ADM)
checkup(ADM)	integrity(ADM)	rcvfile(C)	usemouse(C)
cleanque(ADM)	kbmode(ADM)	rcvprint(C)	uuinstall(ADM)
cleantmp(ADM)	link_unix(ADM)	rcvtrip(C)	uulist(ADM)
cnvtmbox(ADM)	ln(C)†	relax(ADM)	vidi(C)
configure(ADM)	lock(C)	rmuser(ADM)	xbackup(ADM)
cps(ADM)	logs(F)	sar(ADM)†	xdumpdir(ADM)
custom(ADM)†	maildelivery(F)	schedule(ADM)	xrestore(ADM)

In addition, certain chapters describe functionality that is entirely added value. These chapters are listed in Table 1-5.

Table 1-5 Value-added chapters

No.	Name	Description
2	Using the system administration shell	This chapter describes the **sysadmsh**(ADM) menu system, which covers all aspects of system administration.
4	Administering user accounts	The Accounts branch of the **sysadmsh** is described in this chapter, in addition to extensions to the **passwd**(C) command.
9	Maintaining system security	This chapter describes the security features designed to meet the C2 level of trust, which are a feature of this version of the UNIX system.
10	Using the audit subsystem	This chapter describes the audit subsystem that creates security-related log records. This is part of the security features discussed in Chapter 9 (page 201).
14	Setting up electronic mail	An enhanced version of the MMDF mailer is described here.
15	Backing up filesystems	The Backups branch of the **sysadmsh** is described in this chapter, including the **schedule**(ADM) file that controls scheduled backups.
C	Using the system console and color displays	The value-added utilities **kbmode**(ADM), **multiscreen**(M), **vidi**(C), and **setcolor**(C) are described in this chapter.

Chapter 2

Using the system administration shell

The **sysadmsh** (**sys**tem **adm**inistration **sh**ell) is a menu interface designed to simplify the task of system administration. The **sysadmsh** allows you to run the numerous system administration commands with their various options without having to use the traditional UNIX system command line.

This chapter explains how to use the **sysadmsh** interface. To use **sysadmsh** effectively, you also need to know something about the UNIX system commands called by **sysadmsh**. Where appropriate, command-line equivalents to **sysadmsh** menu options are included in the text. However, you should note that some **sysadmsh** options do not have command-line equivalents, for example, none of the **audit** functions do.

You will find it easier to learn the material in this chapter if you start the **sysadmsh** and actually run the examples as you get to them.

This chapter assumes that you have some knowledge of the UNIX operating system. You should become familiar with the concepts covered in the *Tutorial* before using the **sysadmsh** options.

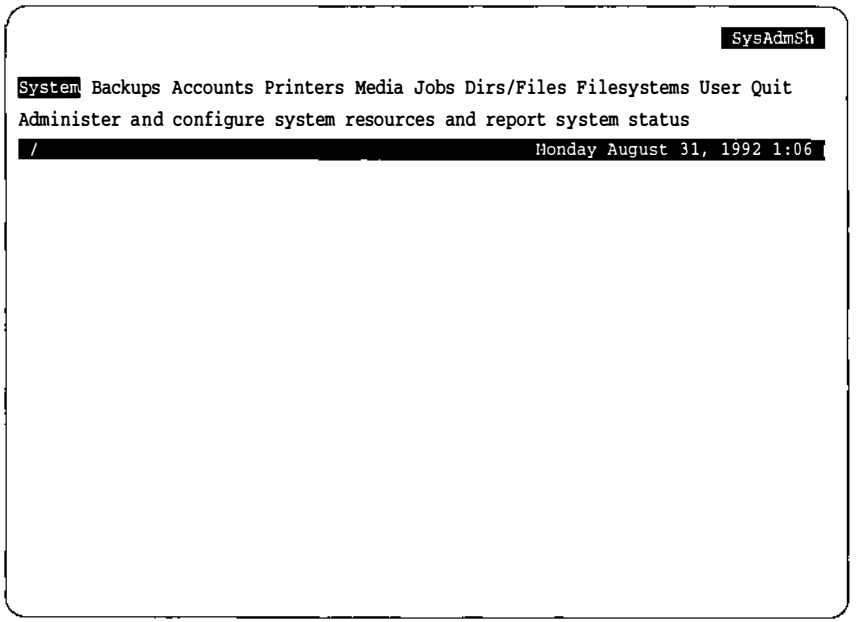
Starting sysadmsh

To gain access to all the functionality of **sysadmsh**, log in as *root* and enter the following command:

sysadmsh

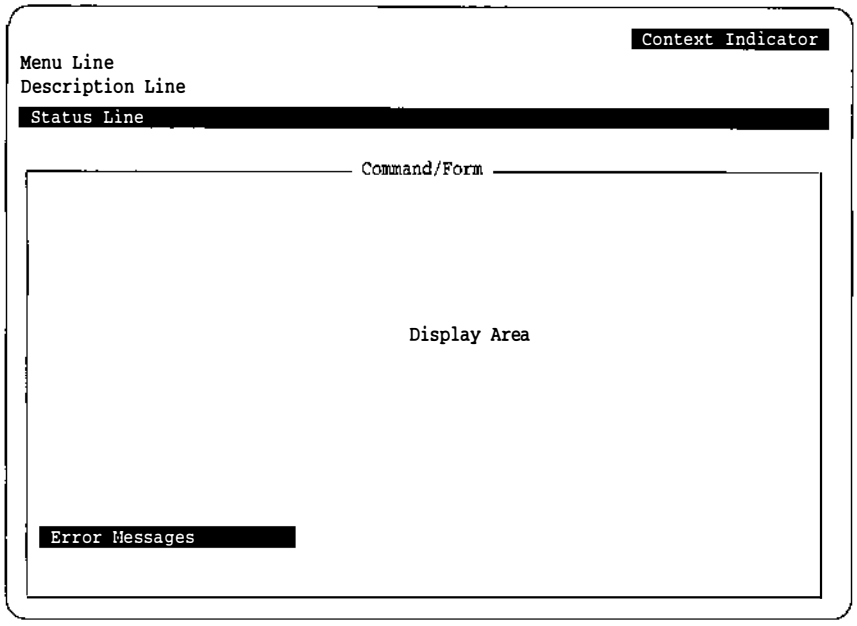
NOTE Only system administrators can make most selections from **sysadmsh**, and only the *root* account can make all selections. If you have insufficient authorization to perform an action (such as backups) you see an error message stating that access is denied because an authorization is required. Refer to "Subsystem authorizations" (page 91) for more information.

The main **sysadmsh** menu is displayed:



How the screen is organized

This is a schematic of the **sysadmsh** screen. Areas shown in black appear on the screen as highlighted areas or bars of text. Each area displays specific types of information:



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

- The Command/Form Line displays a title for the contents of the Display Area. The title can be either a UNIX system command name or the name of a **sysadmsh** form. When a command name is displayed, the location of the manual page associated with the command is appended in parentheses. For example, when System ⇨ Report ⇨ Users is selected, the Command Line displays "`who C`". This means that the command can be found on the **who**(C) manual page.
- The Display Area displays **sysadmsh** forms and scan windows. Forms and scan windows are explained in detail later in this chapter.
- Error Messages and recovery instructions appear on the last line of the screen in highlighted text.

Selecting menu items

The keyboard and mouse operations listed in Table 2-1 are used to move through the menus. Note that there are several ways to select options; if you have used menu-based programs before, use the method you are most familiar with.

Table 2-1 Basic menu operations

Keyboard	Mouse	Action
Arrow keys or ⟨Space⟩ (same as right arrow)	Move cursor to option	moves to menu option.
First letter of option, or move highlight to option and press ⟨Return⟩	Left or middle button	selects menu option.
⟨Esc⟩	Right button	retreats to previous menu.
⟨F1⟩	–	gets help.

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

sysadmsh has a hierarchical menu structure. Many of the menu options move you down to another menu. For example, when you select the Jobs option from the main menu, a submenu containing more options is displayed which lets you check on and manipulate your machine's processes. The menu hierarchy makes it easy to find the command you need by moving down from one menu to the next. Eventually you get to a menu option that either executes a UNIX system command or displays a form that you must fill in with the details that the command needs. Note that typing the first letter of the

option name is the quickest way to move through menu levels; in time you will be able to reach the function you need instantly by pressing three- and four-letter codes you have memorized. (Table 2-9 at the end of this chapter lists all available **sysadmsh** selections and their shorthand forms.)

The best way to learn how to use menus is to practice making menu selections with the keyboard or mouse. If you select an option by mistake, you can always retreat to the previous menu by pressing the ⟨Esc⟩ key or the right mouse button. If you are several levels deep, you can return to the main menu by pressing the ⟨F2⟩ key and then typing **n**. ⟨F2⟩ takes you to the Quit option, and **n** returns you to the main menu. (The ⟨F2⟩ key does not work if you have been dropped into a UNIX system command, such as **vi**(C).) To help you find your way through the **sysadmsh** menus, Table 2-2 contains a map of the second-level menus.

Table 2-2 Map of second-level menus

System	Backups	Accounts	Printers	Media
↓	↓	↓	↓	↓
Report	Create	User	Configure	List
Configure	Restore	Defaults	Schedule	Extract
Hardware	Schedule	Terminal	Request	Archive
Software	View	Report	Auxiliary	Format
Audit	Integrity	Check	Priorities	Duplicate
Execute				Tapedump
Terminate				

Jobs	Dirs/Files	Filesystems	User*	Quit
↓	↓	↓		↓
Report	List	Check		Yes
Terminate	View	Mount		No
Authorize	Copy	Unmount		
	Edit	Add		
	Modify	Floppy		
	Print	DOS		
	Archive			
	Differences			
	Remove			
	UseDOS			

* This selection can be used for user-specific applications defined by the **SA_USERAPPS** environment variable or read from **$HOME/***.sysadmmenu* (see **sysadmsh**(ADM) and **sysadmmenu**(F) for more information).

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

- A lower-level menu is displayed.
- You are dropped into a form.
- A UNIX system command is executed and the result displayed in a scan window.

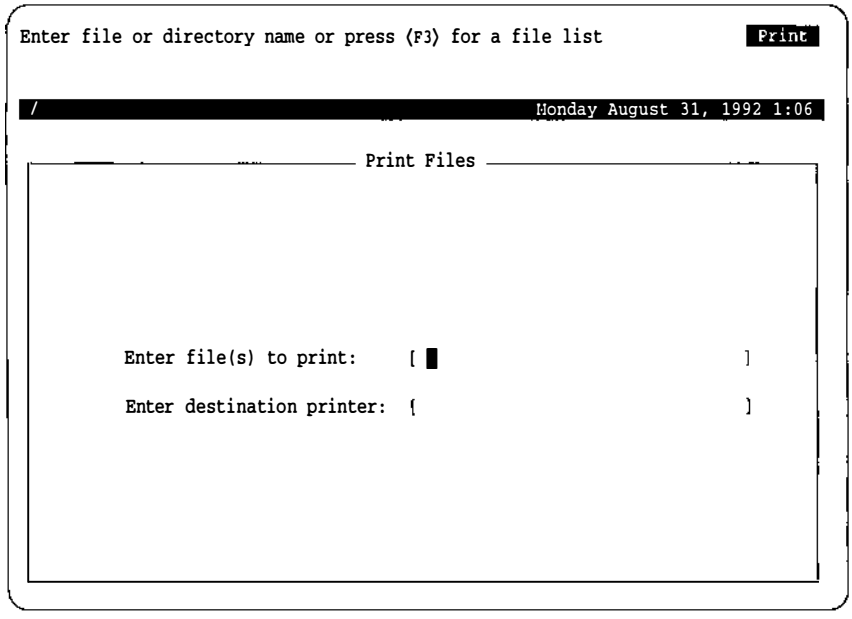
The next two sections explain forms and scan windows.

Using forms

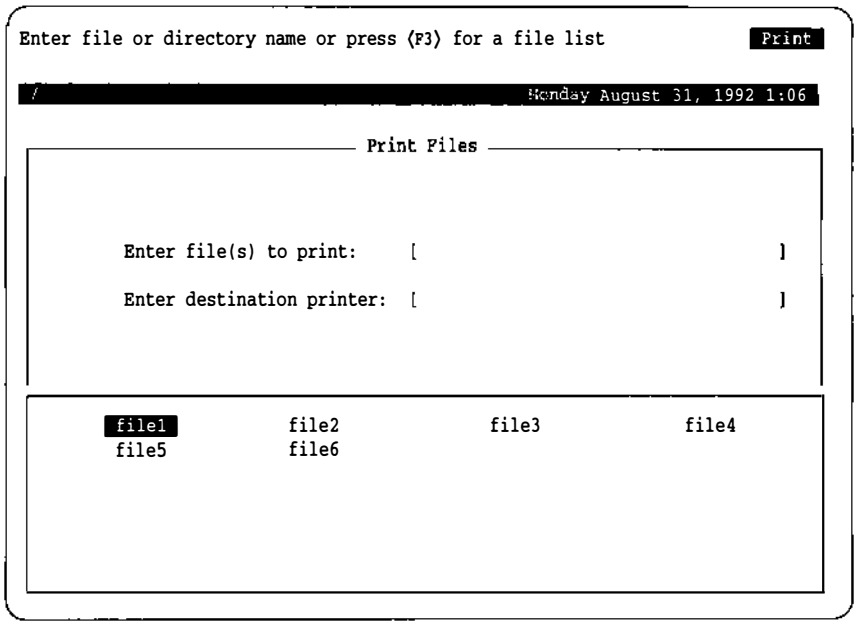
Some menu options require additional information to perform the correct task. For example, the Print option cannot do anything until you tell it what you want to print and which printer to use. When you select this type of option, a form appears on the screen. By filling in the form, you give the command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

The following example demonstrates how forms work, by showing you how to print a file in your current directory. After the example, Tables 2-3, 2-4, and 2-5 list the keystrokes that allow you to move around the form, edit it, and make "point-and-pick" selections.

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



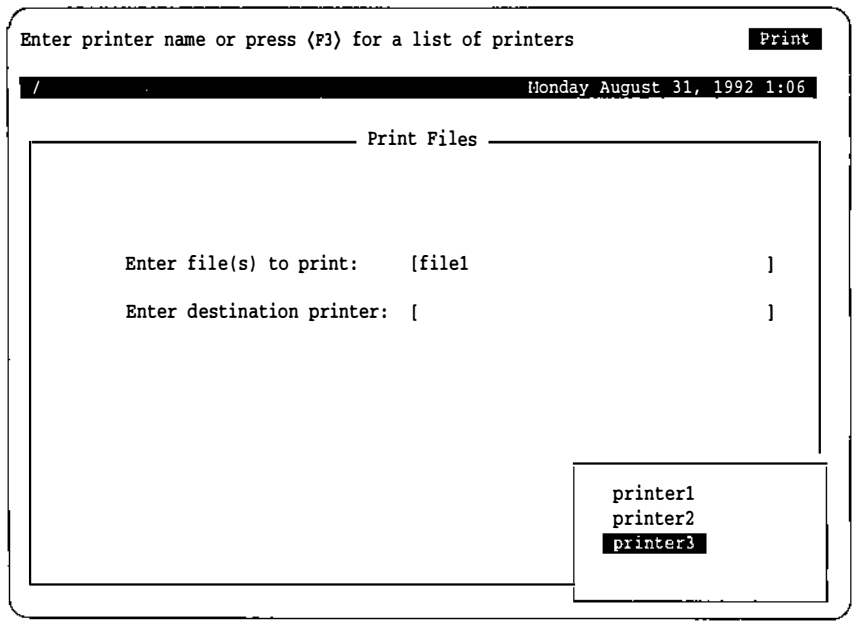
Notice that the highlight is on the first item in the form. You can fill in the field or obtain a list of choices by pressing ⟨F3⟩. You can enter the filename if you know it but, for the sake of this exercise, assume that you need to find the filename and press ⟨F3⟩ now. A window opens up overlapping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



The window contains a list of the files that you can select. To select a file, "point" to it by highlighting it, and "pick" it by pressing return. This is known as point-and-pick, and it is used whenever a range of choices is displayed. After making your selection, the window closes and you return to the Print form.

Note that the name of the file you selected is now displayed in the form. You can now change the name using the edit keys (listed in Table 2-4 later in this section), or press ⟨Return⟩ to move to the next field.

Now enter the name of the printer to be used. If you do not know the printer name, press ⟨F3⟩. Another, smaller window opens that contains a list of installed printers:



You can select the printer just as you did the name of the file. After selecting a printer, you return to the Print menu.

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

Table 2-3 Form operations

Keyboard	Mouse	Action
⟨Esc⟩	Right button	tells the program that you changed your mind and do not want to finish filling in this form. The form is removed, and no action is performed. You are returned to the previous menu. In addition, ⟨Esc⟩ followed by ⟨Return⟩ acknowledges that an error message was read and that you are ready to continue.
Up, Down Arrow	Left button	moves to other fields in a form. Some fields are restricted and no input is allowed. The Arrow keys skip over these. Other fields must be filled in. With the mouse, move the cursor to the field and click the Left button. Pressing the Down Arrow key on the last item in a form brings you back to the first item.
Left, Right Arrow	–	moves left and right in the current field. This allows you to change text without retyping the entire line.
⟨Return⟩	Middle button*	completes the data entry to a field and moves the cursor to the next field. In the last field, pressing ⟨Return⟩ or the middle mouse button completes the entire form and tells the shell that the data is ready to use.
⟨Ctrl⟩x	–	exits and executes the form from wherever you are. Think "x" for "execute". ⟨F10⟩ does the same.
⟨F4⟩	–	calls the spelling checker utility when you are in a form. If you think a word might be misspelled, press ⟨F4⟩ while the cursor is on the word and a list of possible correct spellings appears in a point-and-pick list. The word you select replaces the misspelled word.
⟨F10⟩	–	exits and executes the form from wherever you are. ⟨Ctrl⟩x does the same.

* On a two-button mouse, pressing both buttons simultaneously emulates the middle mouse button.

Table 2-4 Edit keystrokes*

Keystroke	Action
⟨Ctrl⟩**y**	deletes the current line, and begins the line again.
⟨Ctrl⟩**w**	deletes the current word.
⟨Ctrl⟩**g**-⟨Ctrl⟩**h**	moves the cursor to the beginning of the line.
⟨Ctrl⟩**g**-⟨Ctrl⟩**l**	moves the cursor to the end of the line.
⟨Ctrl⟩**v**	toggles into or out of overstrike mode.
⟨Del⟩	deletes the character over the cursor.
⟨Bksp⟩	backs up and deletes one character (left of cursor).
⟨Ctrl⟩**u**	pages up — moves up one page.
⟨Ctrl⟩**d**	pages down — moves down one page.
⟨Ctrl⟩**n**	goes to the next word.
⟨Ctrl⟩**p**	goes to the previous word.
Left, Right Arrow	moves left and right within the edit line.

* You cannot use the mouse for edit operations; if a form opens a window for input, you must use the keyboard. Even the Right button (⟨Esc⟩) will not affect the window.

NOTE The edit keystrokes can also be used to edit the name of the current directory (accessed using ⟨F6⟩).

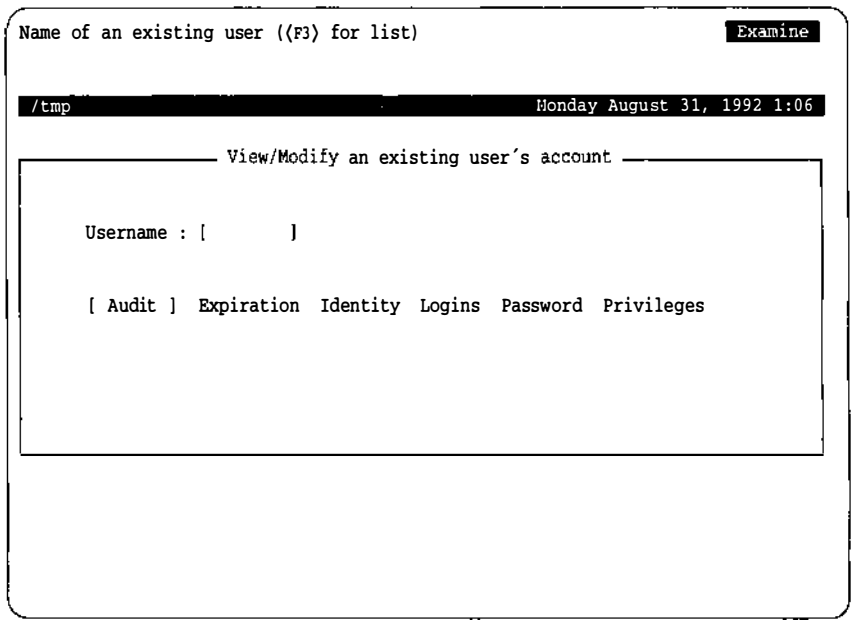
Table 2-5 Point-and-Pick operations

Keyboard	Mouse	Action
⟨Return⟩	Middle button*	selects the item.
⟨Esc⟩	Right button	ends the selection process. The list is removed and no action is performed.
⟨Ctrl⟩v	–	toggles between selecting all or none of the items appearing in a list.
Up, Down Arrow	Move mouse up or down	moves to other items in a list.
Left, Right Arrow	Move mouse right or left	moves across a multicolumn display.
⟨Space⟩	Left button	marks items when the application accepts more than one. A marked item is indicated by an asterisk (*) in the left column. It may be unmarked by pressing ⟨Space⟩ a second time while the item is selected. The entire collection of marked items is selected by pressing ⟨Return⟩.
⟨F5⟩	–	finds items in long listing. It is called the "Search" key. A prompt appears and you enter the string to search for, then press ⟨Return⟩. If the item is found, the highlight moves to that item, and another ⟨Return⟩ selects the item. If no match is found, the highlight does not move. The ";" and ":" keys repeat the previous search, forward and backward respectively.
First letter	–	selects an item by its first letter. It is the fastest method of selection. Pressing ⟨Return⟩ selects the highlighted item. (If there is only one item beginning with that letter, it is marked by typing its first letter. There is no need to press ⟨Return⟩ again.) If several items begin with the same letter, the cursor moves to the first occurrence in the list.

* On a two-button mouse, pressing both buttons simultaneously emulates the middle mouse button.

Using radio buttons

Radio buttons are rows of selection boxes in a form. They are selected in a way similar to items in a menu bar. The purpose of radio buttons is to provide a secondary level of selections from within a form, as in this example, taken from the Accounts menu of the **sysadmsh**:



In this example, the "Username" field must be filled in before the radio button selection can be made. The reference syntax for **sysadmsh** includes a notation for radio buttons, which is a colon (:). Look at the following reference:

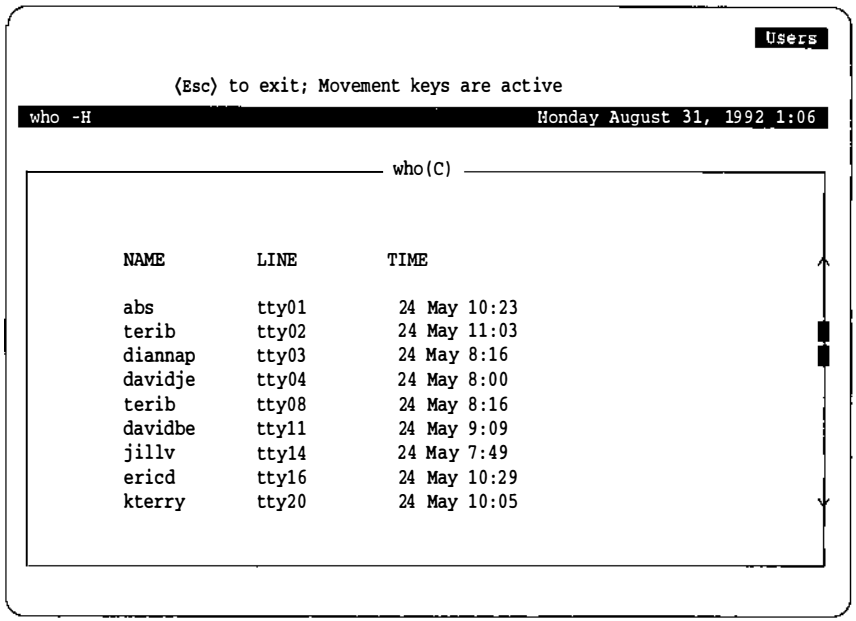
Accounts ⇨ User ⇨ Examine:Audit

When you see a **sysadmsh** reference like this, it means that when you enter the Examine menu you must provide information (in this case, the user name) before making the next selection: Audit.

Using scan windows

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

When you select the Current option, a scan window displaying the output of the **who**(C) command appears in the display area:



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

The vertical scroll bar (at the extreme right edge of the window) shows the position of the current screen relative to the whole document, and moves the screen up or down by an amount that depends on how far you move the scroll bar. When you are viewing the first screen in a document, the top arrow is covered by the highlighted block; when you are viewing the last screen in a document, the bottom arrow is covered by the highlighted block. If the document is less than one page long, the scroll bar does not appear.

To operate the scroll bar, press and hold the ⟨Down Arrow⟩ or ⟨Up Arrow⟩ keys, depending on which direction you want the screen to move over the page. When the scan window shows the area of the document that you want to view, release the key. The ⟨PgUp⟩ and ⟨PgDn⟩ keys can be used to move over the document a page at a time.

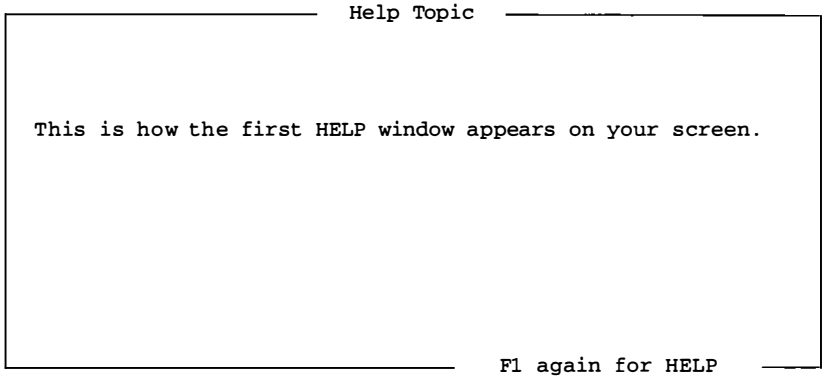
Use the keys listed in Table 2-6 when you are in a scan window.

Table 2-6 Scan operations

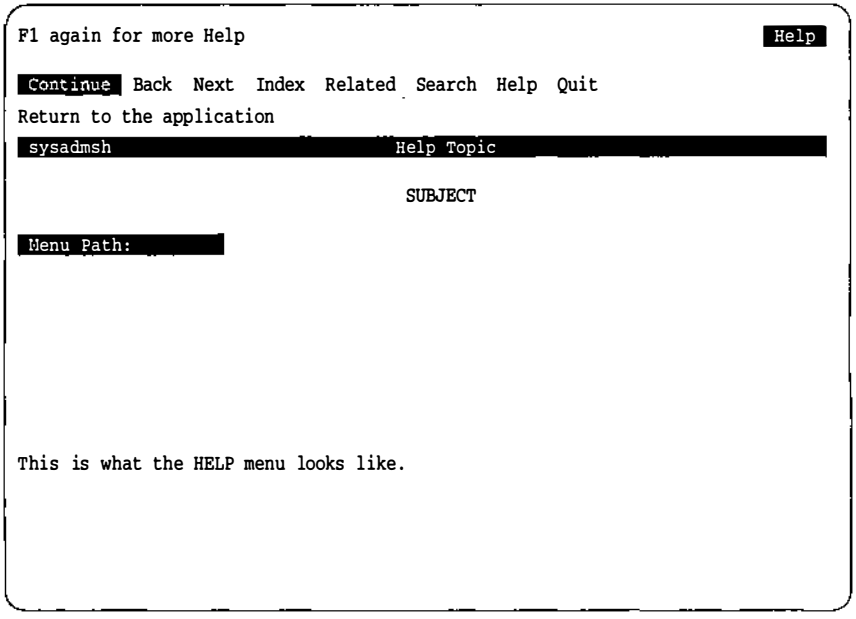
Keyboard	Mouse	Action
⟨Esc⟩	Right button	exits the file.
⟨Up Arrow⟩	–	moves up one line.
⟨Down Arrow⟩ or ⟨Return⟩	–	moves down one line.
⟨PgDn⟩ or ⟨Space⟩	Middle button	moves down a page.
⟨PgUp⟩	Left button	moves up a page.
⟨Home⟩	–	moves to the top of the display.
⟨End⟩	–	moves to the bottom of the display.
⟨F5⟩	–	searches for a pattern in the display. (The ";" and ":" keys repeat the search forward and backward, respectively.)
⟨F7⟩	–	prints the output of the command or file currently in the scan window.

Getting help

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



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



When you are finished, select Quit from the Help menu and you return to your place in the **sysadmsh** menu.

The menu options for Help are listed in Table 2-7.

Table 2-7 Help options

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

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

NOTE When you are within a particular UNIX system command, you do not have access to the Help facility. For example, when you select Dirs/Files ➪ Edit, you are within the UNIX system **vi** command, and the **sysadmsh** keys no longer function. When you exit the command and return to the **sysadmsh**, the keys function as expected. If no element of the **sysadmsh** is visible on the screen (menu line, boxes, context indicator, and so on) then Help is probably unavailable. If you need help, exit from the current process and press the ⟨F1⟩ key to view Help. In general, it is best to use Help prior to executing a menu selection.

Changing the current directory within sysadmsh

There are many occasions when it is necessary to change your current directory to use certain files and commands. You can move to another directory by pressing the ⟨F6⟩ key. The current directory is displayed at the top of the screen. You can use the ⟨Bksp⟩ key to erase the name of the current directory (to begin again), or you can add to or alter part of the current name using the edit keystrokes (Table 2-2). When you press ⟨Return⟩, your directory change is executed and reflected on the status line.

The function keys

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

Table 2-8 Function keys

Key	Action
⟨F1⟩ Help key	displays help for the current context within the application. Further information is available by pressing ⟨F1⟩ again.
⟨F2⟩ Exit key	activates the Quit option on the top menu level. Press **n** to return to **sysadmsh** or **y** to exit.
⟨F3⟩ Pop-up key	(used within a form) displays a list of items that are acceptable for the current field.
⟨F4⟩ Spell key	(used within a form) displays a list of words that are possible correct spellings of the word in the current field. Select a word from the list by highlighting it, then pressing ⟨Return⟩. The word is then placed in the field. This function is only available if the **Speller** is installed.

(Continued on next page)

Table 2-8 Function keys
(Continued)

⟨F5⟩ Search key	(used within a window) prompts for a string to search for. When you enter a string and press ⟨Return⟩, the highlight moves to the item in the list that matches the pattern. If no match is found, the search fails and the highlight does not move. In addition, the semicolon (;) repeats a search forward and the colon (:) searches backward.
⟨F6⟩ New directory key	offers the opportunity to change your current working directory. Note that this does not change the directory you will return to upon leaving **sysadmsh**.
⟨F7⟩ Print key	prints the output of any command that is displayed in a scan window.

Using shell escapes to access the UNIX system command line

You can execute a UNIX system command from within a **sysadmsh** menu by typing the shell-escape character, an exclamation point or mark (!). The menus are replaced by a subshell that displays a text-entry line and a prompt asking for a command. When you enter the command and press ⟨Return⟩, the command is executed by the shell. After the command is completed, the output is displayed on the screen, and you are prompted to press any key to return to the shell.

NOTE The UNIX system command line can only be accessed from the shell menus. It cannot, for example, be accessed from a form or a point-and-pick list.

sysadmsh environment variables

The **sysadmsh** uses the following environment variables, which can be defined in user *.login* or *.profile* files:

SA_EDITOR	If not set, the default editor is Lyrix if installed, or **vi**(C) if Lyrix is not available.
SA_PRINT	If not set, **sysadmsh** pipes output to the */bin/lp* program; if set, **sysadmsh** sends output directly to the */dev/lp* printer device.
SA_USERAPPS	If not set, the default user application file is **$HOME/***.sysadmmenu.*

Customizing sysadmsh menus

Third party developers can customize **sysadmsh** menus as desired, including changing colors. See **menumerge**(ADM) and **sysadmcolor**(F) for complete instructions.

sysadmsh menu options

Table 2-9 lists the options available in the **sysadmsh** menus. The main or top-level menu options appear in boldface type at the left margin of the table. The options listed below each top-level menu option are in lower levels of the menu hierarchy. Those with one arrow are in the first-level menus, those with two arrows are in the second-level menus, and so on. For instance, at the beginning of the table, Report is a first-level menu option in the System menu, and Activity is an option in the second-level Report menu.

Once you are familiar with the menu options, you can enter shorthand menu paths to reach lower level options without having to see all of the upper level menus. In Table 2-9, the "Path" column lists the shorthand menu path for each menu option. Enter the first letter of each menu option in the path down to the option you want. For example, to choose the Activity option, enter the first letters of system, report, and activity, **sra**, without pressing the ⟨Return⟩ key.

If there are no menu options listed below the one you choose, either a form or a display appears. For example, when you choose the Activity option, a scrollable display or scan window appears, showing system processes.

Table 2-9 sysadmsh menu map

sysadmsh option	Path	Function
System	s	system-wide reports, configurations
➪Report	sr	reports on current state of system
➪➪Activity	sra	reports on current system activity
➪➪Users	sru	reports users currently logged in
➪➪Printers	srp	reports on current status of printers
➪➪Disk	srd	reports on current disk usage
➪➪Network	srn	reports on current network files
➪➪➪Xnet	srnx	reports on XENIX-NET status
➪➪➪UUCP	srnu	reports on UUCP status
➪➪Messages	srm	reads system messages
➪➪Software	srs	checks installed software/package status
➪Configure	sc	configures system files
➪➪Security	scs	changes level of system security
➪➪➪Relax	scsr	resets security to default UNIX system levels
➪➪Kernel	sck	changes kernel parameters or capabilities
➪➪➪ParaMeters	sckp	configures tunable kernel parameters
➪➪➪Rebuild	sckr	relinks kernel according to current settings
➪➪➪DOS	sckd	adds DOS filesystem support to the kernel
➪➪➪Streams	scks	adds streams support to the kernel
➪➪➪Layers	sckl	adds shell layers support to the kernel
➪➪Logout	scl	sets idle time before user is logged out
➪➪Defaults	scd	modifies system default parameters
➪➪➪Home	scdh	modifies default home directory for users
➪➪➪Message	scdm	modifies message of the day file
➪➪➪Checklist	scdc	lists which filesystems checked at startup
➪➪➪Other	scdo	modifies other default files
➪➪International	sci	configures system for international use
➪➪➪System	scis	sets system locale variables
➪➪➪Individual	scii	changes a user's default locale
➪➪➪Display	scid	changes mapping of terminal character set
➪➪➪Keyboard	scik	changes mapping of console keyboard
➪➪Network	scn	configures networking files
➪➪➪UUCP	scnu	configures UUCP files, enable/disable tty
➪➪Time	sct	sets the system time
➪➪Menus	scm	customizes **sysadmsh** menus
➪➪Other	sco	executes third-party **sysadmsh** extensions
➪Hardware	sh	adds or removes hardware from the system
➪➪HardDisk	shh	adds a hard disk to the system
➪➪Tape	sht	adds or removes a tape drive from system

(Continued on next page)

Table 2-9 sysadmsh menu map
(Continued)

sysadmsh option	Path	Function
⇨⇨Printer	shp	adds or removes a printer from system
⇨⇨Card_Serial	shs	adds a serial card to system
⇨⇨Mouse	shm	adds mouse to system
⇨⇨Video	shv	configures video card graphics parameters
⇨Software	ss	adds or removes software from system
⇨Audit	sa	administers/examines system auditing data
⇨⇨Enable	sae	enables audit using existing parameters file
⇨⇨Disable	sad	disables audit (stops collection of process data)
⇨⇨Collection	sac	displays or modify audit collection rules
⇨⇨⇨Directories	sacd	displays or modify audit directory list
⇨⇨⇨⇨List	sacdl	displays audit directory list
⇨⇨⇨⇨Create	sacdc	creates new audit directory
⇨⇨⇨⇨Delete	sacdd	deletes existing audit directory
⇨⇨⇨⇨Add	sacda	adds entry to audit directory list
⇨⇨⇨⇨Remove	sacdr	removes entry from audit directory list
⇨⇨⇨Events	sace	displays/modifies list of audited events
⇨⇨⇨⇨View	sacev	displays system audit collection mask
⇨⇨⇨⇨Modify	sacem	modifies system audit collection mask
⇨⇨⇨IDs	saci	displays/modifies list of users/groups audited
⇨⇨⇨⇨View	saciv	displays list of users and groups audited
⇨⇨⇨⇨Modify	sacim	modifies list of users and groups audited
⇨⇨⇨Parameters	sacp	displays or modifies audit parameters
⇨⇨⇨⇨View	sacpv	displays audit parameters
⇨⇨⇨⇨Modify	sacpm	modifies audit parameters
⇨⇨⇨Reset	sacr	changes collection rules to default values
⇨⇨⇨Statistics	sacs	displays statistics of current audit session
⇨⇨Report	sar	reports on stored audit session data
⇨⇨⇨List	sarl	lists report templates available
⇨⇨⇨View	sarv	views parameters in a report template
⇨⇨⇨Create	sarc	creates a new report template
⇨⇨⇨Modify	sarm	modifies an existing report template
⇨⇨⇨Delete	sard	deletes an existing report template
⇨⇨⇨Generate	sarg	generates report of an audit session
⇨⇨Files	saf	manipulates audit session files
⇨⇨⇨List	safl	lists audit session files on system

(Continued on next page)

Table 2-9 sysadmsh menu map
(Continued)

sysadmsh option	Path	Function
➪➪➪Backup	safb	backs up audit session file to removable media
➪➪➪Delete	safd	removes audit session file
➪➪➪Restore	safr	restores audit file from removable media
➪Execute	se	executes programs that are system specific
➪Terminate	st	shuts down system to remove power or reboot
Backups	b	performs backups of files
➪Create	bc	creates backups
➪➪Scheduled	bcs	performs scheduled filesystem backups
➪➪Unscheduled	bcu	performs unscheduled filesystem backup
➪Restore	br	restores filesystems and files
➪➪Partial	brp	restores specific directories and files
➪➪Full	brf	restores entire filesystem
➪Schedule	bs	modifies scheduled backup frequency
➪View	bv	views contents of a backup
➪Integrity	bi	checks the integrity of a backup
Accounts	a	controls functions of user accounts
➪User	au	alters/creates/retires user accounts
➪➪Examine	aue	sets password/ID/authorization/audit parameters
➪➪Create	auc	makes a new user account
➪➪Retire	aur	closes an existing user account
➪Defaults	ad	sets system-wide (default) parameters
➪➪Authorization	ada	views/modifies default kernel/subsystem privileges
➪➪Password	adp	views/modifies default password life/choice
➪➪Logins	adl	views/modifies default login attempt controls
➪Terminal	at	manages terminal database entries
➪➪Examine	ate	views/modifies existing terminal entry
➪➪Create	atc	makes a new terminal entry
➪➪Delete	atd	deletes an existing terminal entry
➪➪Lock	atl	locks a specific terminal
➪➪Unlock	atu	clears all locks on a specific terminal
➪➪Assign	ata	manages device name equivalences database

(Continued on next page)

Table 2-9 sysadmsh menu map
(Continued)

sysadmsh option	Path	Function
➪➪➪Examine	atae	views/modifies existing device entry
➪➪➪Create	atac	makes a new device entry
➪➪➪Delete	atad	deletes an existing device entry
➪Report	ar	reports on passwords/terminals/login activity
➪➪Password	arp	reports on accounts by password status
➪➪➪Impending	arpi	reports on accounts: passwords near expiring
➪➪➪Expired	arpe	reports on accounts with expired passwords
➪➪➪Dead	arpd	reports on accounts with dead passwords
➪➪➪User	arpu	reports on single user's password
➪➪➪Group	arpg	reports on single group's passwords
➪➪➪Full	arpf	lists all entries in password database
➪➪Terminal	art	reports on access status by terminals
➪➪Login	arl	reports login activity by user/group/terminal
➪➪➪User	arlu	reports logins of one, a range of, or all users
➪➪➪Group	arlg	reports logins of a group or range of groups
➪➪➪Terminal	arlt	reports logins of one/range/all terminals
➪Check	ac	checks contents of tcb files for errors
➪➪Databases	acd	checks consistency of subsystem databases
➪➪Password	acp	checks */etc/passwd* and */etc/group*
Printers	p	administers print system
➪Configure	pc	configures printers on print service
➪➪Add	pca	adds printer to system
➪➪Modify	pcm	modifies printer configuration
➪➪Remove	pcr	removes printer destination from print service
➪➪Default	pcd	changes system default destination printer
➪➪Parameters	pcp	modifies printer controls and parameters
➪➪Errors	pce	sets error warning notification/recovery modes
➪➪Content	pcc	specifies type of content printable on printer
➪➪Users	pcu	specifies who can use printer
➪Schedule	ps	starts/stops print service, handle requests
➪➪Begin	psb	starts print service
➪➪Stop	pss	shuts down print service
➪➪Accept	psa	allows requests for destination

(Continued on next page)

Table 2-9 sysadmsh menu map
(Continued)

sysadmsh option	Path	Function
➪➪Reject	psr	rejects requests for destination
➪➪Enable	pse	enables printers
➪➪Disable	psd	disables printers
➪Request	pr	moves or cancels requests on print service
➪➪Move	prm	moves requests between destinations
➪➪Cancel	prc	cancels requests made to print service
➪Auxiliary	pa	handles print-wheels/filters/preprinted forms
➪➪Alert	paa	sets/lists an alert for print-wheel
➪➪Filter	paf	administers filters used with print service
➪➪➪Change/Add	pafc	adds or changes filter used with print service
➪➪➪Remove	pafr	removes filter from print service
➪➪➪List	pafl	lists a description of filter
➪➪➪Original	pafo	restores original filter description
➪➪PPforms	pap	administers pre-printed forms of print service
➪➪➪Configure	papc	modifies printer settings for pre-printed forms
➪➪➪Modify	papm	adds or changes pre-printed forms
➪➪➪Remove	papr	removes pre-printed form from print service
➪➪➪List	papl	lists attributes of existing form
➪➪➪Users	papu	allows or denies user access to a form
➪➪➪Alerts	papa	modifies alert method for mounted form
➪➪➪➪Specify	papas	specifies alerting method
➪➪➪➪List	papal	lists current alert
➪➪➪➪Terminate	papat	terminates existing active alert
➪➪➪➪Remove	papar	removes alert definition
➪Priorities	pp	sets printing queue priorities
➪➪Default	ppd	sets system-wide priority default
➪➪Highest	pph	sets default highest priority level for users
➪➪Remove	ppr	removes users from specified priority level
➪➪List	ppl	lists default priority level and limits

(Continued on next page)

Table 2-9 sysadmsh menu map
(Continued)

sysadmsh option	Path	Function
Media	m	reads, copies, compares, formats floppies/ tapes
➪List	ml	lists the contents of a floppy or tape
➪Extract	me	extracts the contents of a floppy or tape
➪Archive	ma	stores files/directories/filesystems on media
➪Format	mf	formats either a UNIX or DOS floppy
➪Duplicate	md	makes a copy of floppy or tape
➪Tapedump	mt	displays physical contents of tape
Jobs	j	views/controls processes
➪Report	jr	reports on current processes (snapshot)
➪Terminate	jt	terminates currently running process (kill)
➪Authorize	ja	authorizes users to run jobs
➪➪Scheduled	jas	authorizes creating regularly scheduled jobs
➪➪➪Default	jasd	sets default authorization for scheduled jobs
➪➪➪User	jasu	allows/prohibits user-scheduled jobs
➪➪➪View	jasv	checks who can create scheduled jobs
➪➪Delayed	jad	authorizes users to create delayed jobs
➪➪➪Default	jadd	sets the default authorization for delayed jobs
➪➪➪User	jadu	allows/prohibits user from creating delayed jobs
➪➪➪View	jadv	checks who is authorized to create delayed jobs
➪➪Environment	jae	modifies delayed job environment
➪➪➪At	jaea	modifies the environment for at-controlled jobs
➪➪➪Batch	jaeb	modifies the environment for **batch**(C) jobs
Dirs/Files	d	interacts with files and directories
➪List	dl	lists files in current directory
➪View	dv	views contents of file
➪Copy	dc	copies directory or file
➪Edit	de	edits one or more files
➪Modify	dm	changes file parameters
➪➪Permissions	dmp	changes file permissions
➪➪Ownership	dmo	changes file ownership

(Continued on next page)

Table 2-9 sysadmsh menu map
(Continued)

sysadmsh option	Path	Function
➪➪Group	dmg	changes file group ownership
➪➪Name	dmn	renames or moves files
➪➪Size	dms	compacts files
➪➪Format	dmf	changes file formats
➪Print	dp	prints files
➪Archive	da	stores files
➪Differences	dd	compares two text files or directories
➪Remove	dr	removes specified files or directories
➪UseDOS	du	uses DOS utilities to manipulate DOS files
➪➪List	dul	lists DOS files in current directory
➪➪Remove	dur	removes a DOS file or directory
➪➪MakeDir	dum	creates a DOS directory
➪➪Copy	duc	copies files between DOS and UNIX systems
➪➪View	duv	displays DOS files
➪➪Format	duf	formats DOS media
Filesystems	f	checks/acts on filesystems
➪Check	fc	checks and repairs filesystem (**fsck**)
➪Mount	fm	mounts a filesystem
➪Unmount	fu	unmounts a mounted filesystem
➪Add	fa	adds appropriate info for a new filesystem
➪Floppy	ff	creates a filesystem on a floppy
➪DOS	fd	adds support for DOS filesystem
User	u	user-specific applications read from *$HOME/.sysadmmenu*
Quit	q	quits system administration shell
➪Yes	qy	leaves the sysadm shell
➪No	qn	cancels Quit command

Chapter 3

Starting and stopping the system

This chapter explains how to do the following:

- Start and stop your system.
- Log in as the super user (*root*).
- Change the system boot procedure.
- Use the device and system configuration information displayed at boot time.

Additional information on customizing the system startup process is found in the "Customizing system startup" appendix in this guide.

Starting the system

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

- Load the operating system.
- Check the filesystems (if the system was improperly stopped).
- Choose the mode of system operation.

The following sections describe each of these procedures.

Loading the operating system

The first step in starting the system is to load the operating system from the computer's hard disk.

1. Turn on power to the computer and hard disk. The computer loads the UNIX system bootstrap program and displays this message:

```
SCO System V/386

Boot
:
```

2. Press the ⟨Return⟩ key. The bootstrap program loads the operating system.

When the system is loaded, it displays information about itself and verifies that the root filesystem (that is, all files and directories) is in order and not corrupted. If a filesystem is uncorrupted and in good order, it is called "clean". If the root filesystem is clean, you can choose the mode of operation. If not, the system requires you to clean the filesystem before choosing.

Cleaning filesystems

You must clean the root filesystem if the following message is displayed:

```
fsstat: root filesystem needs checking
OK to check the root filesystem (/dev/root) (y/n)?
```

This message is displayed only if the system was not stopped properly, as described in "Stopping the system" (page 52).

Each filesystem generates a similar message. In order to work properly, the operating system requires clean filesystems. If the above message does not appear, your filesystem is clean and ready to use.

NOTE When autobooting (see "Booting automatically" (page 56)), the default system behavior is to fix filesystems without operator intervention; the system boots and runs **fsck** assuming "yes" answers. To change this, change **FSCKFIX=YES** to **FSCKFIX=NO** in */etc/default/boot*.

To clean the filesystem, enter **y** (for "yes") and press the ⟨Return⟩ key. The **fsck**(ADM) utility cleans the filesystem, repairing damaged files or deleting files that cannot be repaired. It reports on its progress as each step is completed. At some point, you may be asked if you wish to salvage a file. Always answer by entering **y** or **n** and pressing the ⟨Return⟩ key. For an explanation of how **fsck** works, refer to "Repairing filesystems with fsck" (page 117). for more information.

When cleaning is complete, the system asks you to choose the mode of operation.

Choosing the mode of system operation

You may choose the mode of operation as soon as you see the message:

```
INIT: SINGLE USER MODE

Type CONTROL-d to continue with normal startup,
(or give the root password for system maintenance):
```

The system has two modes: normal operation and system maintenance. Normal operation is for ordinary work on the system. This is the mode that allows multiple users to log in and begin work. It is also known as multiuser mode. System maintenance mode is reserved for work to be done by the system administrator, and does not allow multiple users. It is also known as single user mode.

To choose normal operation, press ⟨Ctrl⟩**d**. The system displays a startup message, and you are prompted to enter the system time (see the next section). Then the system executes commands found in the */etc/rc* directories, (this includes the */etc/rc.d* and */etc/rc2.d* directories, and so forth, referred to collectively as */etc/rc* scripts) generating startup messages for the various system services, such as the printer or network services. (These scripts are described later in this chapter.) Next, the system displays the "login:" prompt. You can now log in as a normal user, as described in the "Logging in, logging out" chapter of the *Tutorial*, or as the super user.

To choose system maintenance mode, enter the super user password (also called the "root password") and press ⟨Return⟩.

NOTE The super user (*root*) password is assigned during system installation. If you do not know the *root* password, ask the administrator who installed your system.

The super user prompt "#" is displayed. The commands in the */etc/rc* scripts are not executed. (Choose system maintenance mode only if you must do system maintenance work that requires all other users to be off the system.) When you log out of system maintenance mode using ⟨Ctrl⟩**d**, the system automatically enters normal operation.

Entering system maintenance mode by shutting down first

To go from normal operation to system maintenance mode, log in as *root* and give the following command to shut down the system:

/etc/shutdown -g*n*

Δ **sysadmsh** users select: System ⇨ Terminate

where ***n*** is the number of minutes until multiuser mode is stopped. After ***n*** minutes has elapsed, you are asked to confirm your choice and the system is shut down. You are then asked to press any key to reboot. When the system has rebooted, give the *root* password to enter single user mode (see "Choosing the mode of system operation" (page 45) for more information).

Entering system maintenance mode directly

To go from normal operation to system maintenance mode directly, log in as *root* and give the following command:

/etc/shutdown -g*n* su

where ***n*** is the number of minutes until multiuser mode is stopped. The **su** indicates that you want to go directly into single user mode.

NOTE There is no **sysadmsh** equivalent for this command.

Setting the time and date

Once normal operation starts, the system asks for the correct time and date. It displays the current time and date and then the following message:

```
INIT: New run level: 2

Current System Time is Wed Nov 13 08:19:00 PST 1991
Enter new time ([yymmdd]hhmm):
```

Unless your clock battery is drained or removed, there should be no need to change the date. To leave the time and date unchanged, simply press ⟨Return⟩. If you need to change the time and date, enter the new time and press ⟨Return⟩. The new values must be entered as two or more consecutive pairs of digits, where the digits may be one or more of the following:

yy (optional) represents the current year. It may be any two-digit value, from 70 to 99 for the years 1970 to 1999, respectively.

mm (optional) represents the current month. It may be any two-digit value, from 01 to 12 for January to December, respectively.

dd (optional) represents the current day. It may be any two-digit value, from 01 to the last day of the month.

hh represents the current hour. It may be any two-digit value, from 00 to 23. Hours are expressed in military time, where morning hours range from 00 to 11 and evening hours from 12 to 23.

mm represents the current minutes. It may be any two-digit value, from 00 to 59.

For example, to change the time and date to February 3, 1991 at noon, enter:

9102031200

Press ⟨Return⟩. After accepting the new value, the system then displays the new time and date:

```
Sun Feb 03 12:00:00 PST 1991
```

If you enter an incorrect value, the system prompts you to try again. If you do not enter an optional value, the current value for that item remains unchanged. If you type a new value for the year, you must also type values for the month and day. Similarly, if you type a new value for the month, you must type a value for the day.

The time and date display is followed by service startup messages and the "login:" message.

Checking the security databases

Each time your system is rebooted (and after **fsck** is run if your system was brought down unexpectedly) the system automatically checks critical security database files. The messages appear as follows:

```
Checking tcb ...
Checking auth database ...
Checking protected subsystems database ...
Checking ttys database ...
```

This checking is done to avoid problems with gaining access to your system. In the rare case where a file is missing, you are alerted to this fact and asked to restore the file from backups.

When the system is halted suddenly by power or hardware failures, some file-system 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, and before entering multiuser mode.

This check proceeds as follows:

1. The script */etc/smmck* (system maintenance mode checker) runs the **tcbck**(ADM) program to clean up any database files that were left in an interim state while being updated.

   When a security database file is updated, the contents of the old file (***file***) is copied or updated to create the new "-t" file (***file-t***). Next, the old file (***file***) is moved to a "-o" file (***file-o***), and the new file (***file-t***) is moved to the original name (***file***). When this process is interrupted, "-o" and "-t" files are left and must be reconciled before the system will function properly. **tcbck** first resolves any "-t" and "-o" files left in the */etc/auth/system*, */etc/auth/subsystems*, */tcb/files/auth/** directories, and the */etc/passwd* and the */etc/group* files. If there are multiple versions of a file, the extra files must be removed. This is done automatically as follows:

   A. If ***file***, ***file-o***, and ***file-t*** exist and ***file*** is not zero length, then ***file-t*** and ***file-o*** are removed.

   B. If ***file*** and ***file-t*** exist then ***file-t*** is removed.

   C. If only ***file-t*** exists, then it is moved to ***file***.

   D. If only ***file-o*** exists, then it is moved to ***file***.

   If scenario C occurs, a message similar to the following is displayed:

   `/etc/tcbck: file` ***file*** `missing, saved` ***file***`-t as` ***file***

   This is done because the "-t" file is the modified version of the original file and could have been damaged; it is likely that this file does not contain all the entries of the original. This message is repeated for all files found in that state in the specified directories. (The "-o" files are not suspect because they are the original versions of the files renamed prior to updating.)

   WARNING If you do not have backups and the files */etc/group* and */etc/passwd* were regenerated by moving the "-t" files, do not restore the original files from your distribution media. The "-t" files will have most (if not all) of your entries, and the distribution versions will not have any.

2. Next, **tcbck** checks that key system files are present and that they are not of zero length. If a file is missing (or zero length), then a message similar to that shown below is displayed:

   `/etc/tcbck: file` ***file*** `is missing or zero length`

This process is repeated for each of the following files:

/etc/auth/system/default†
/etc/auth/system/files
/etc/auth/system/devassign
/etc/auth/system/authorize†
/tcb/files/auth/r/root†
/etc/group
/etc/passwd†

When this process is complete, if any files were missing, or empty "-t" files were substituted for real files, the following message is displayed:

```
/etc/smmck: restore missing files from backup or distribution.
```

NOTE Corrupted files are not detected by **tcbck**, but other error messages are displayed; these messages are described in "Security system troubleshooting" (page 595).

3. If critical database files are removed or corrupted (files marked with a dagger (†)), then the system enters maintenance mode automatically without asking for the root password. The messages appear as follows:

```
INIT: SINGLE USER MODE
Security databases are corrupt.
Starting root shell on console to allow repairs.
Entering System Maintenance Mode
```

If no critical database files are missing, you are prompted to choose system maintenance mode or normal operation.

4. If you find that files must be restored, your first option is to restore the files from your backups. The second option is to restore the files from the UNIX system distribution media.

For example, if the system reported that the file */etc/group* was missing and you had a backup of the root filesystem that was created using **sysadmsh**(ADM), you would use the Backups ➪ Restore ➪ Partial selection and restore the file. If backups are unavailable, you will have to use the distribution files.

† critical tcb files

NOTE If you must restore the file */tcb/files/auth/r/root* from your distribution, retrieve volume N2 and insert it into your floppy drive, then enter the following commands:

```
cd /tcb/files/auth/r
mount -r /dev/install /mnt
cp /mnt/tcb/files/auth/r/root .
```

5. Use **custom**(ADM) to restore any missing files. You can restore only one file at a time; do not try to specify more than one file. When you have restored all the necessary files, exit **custom** and press ⟨Ctrl⟩**d**. You are then prompted to enter system maintenance mode or normal operation. Press ⟨Ctrl⟩**d** again.

6. **tcbck** then removes the files */etc/auth/system/pw_id_map* and */etc/auth/system/gr_id_map* because the modification times of these files are compared with those of */etc/passwd* and */etc/group*, and problems can occur when the system clock is reset. **tcbck** then tries to rebuild the map files using **cps**(ADM). If this fails then either the File Control database (*/etc/auth/system/files*) is missing, or the the File Control database entry for "/" is missing, or there are syntax errors in */etc/passwd* or */etc/group*.

7. After the system enters multiuser mode ("`INIT: New run level: 2`" is displayed) and you are prompted to set the system clock, */etc/authckrc* is reinvoked. If any missing files are found, warnings similar to the ones shown previously are displayed, followed by the message shown below:

   ```
   /etc/tcbck: file file is missing or zero length
   /etc/authckrc: Log in on the OVERRIDE tty and restore
   the missing files from a backup or the distribution disks.
   ```

   This means that you missed some files earlier. These files will have to be replaced when the system comes up in multiuser mode and you are allowed to log in. Write down the names of the missing files at this stage.

8. Next, the following message is displayed:

   ```
   Checking auth database ...
   ```

   The **authck**(ADM) program is run to make certain that all users listed in */etc/passwd* have Protected Password database entries. If any are missing, they are created as needed.

9. Next, the following message is displayed:

   ```
   Checking protected subsystem database ...
   ```

   The **authck**(ADM) program is run. The Protected Subsystem database files are checked to ensure that they correctly reflect the subsystem authorization entries in the Protected Password database. Each name listed in each subsystem file is verified against the Protected Password entry with the same name, so that no authorization is inconsistent

between the files. Also, each Protected Password entry is scanned to verify that all the privileges listed are reflected in the Protected Subsystem database. If any inconsistencies are found, you are asked if you want them fixed automatically:

```
There are discrepancies between the databases.
Fix them (Y or N)?
```

The error messages are found in the **authck**(ADM) manual page.

> **NOTE** If the system is autobooting ("AUTOBOOT=YES" appears in */etc/default/boot*), then **authck**(ADM) is called noninteractively. Warnings are displayed about inconsistencies found but **authck** is not given the opportunity to fix them. The transition to the multiuser operation then proceeds as normal.
>
> You can also run **authck** manually; see the manual page for syntax.

10. Next, you see the following message:

    ```
    Checking ttys database ...
    ```

    ttyupd(ADM) is run to ensure that all ttys in */etc/inittab* have entries in the Terminal Control database (*/etc/auth/system/ttys*).

11. The system is now up and ready for logins. If any files were reported missing, you must now log in on the override terminal to restore them, following the same procedure outlined earlier. By default, the override terminal is defined as *tty01*, also known as the first multiscreen. If you removed the default entry in */etc/default/login*, you will have to shut the system off, reboot and enter single-user mode, and restore the files that way. When you log in on the override tty, the following message is displayed:

    ```
    The security databases are corrupt.
    However, root login at terminal tty01 is allowed.
    ```


Logging in as the super user

Many system maintenance tasks, when performed during normal operation, require you to log in as the super user. For example, you must be logged in as the super user to stop the system.

To log in as the super user, you must know the super user password. You also need to see the "login:" message on your terminal's screen. If you do not see this message, press ⟨Ctrl⟩**d** until it appears.

NOTE The super user *(root)* password is assigned during system installation. If you do not know the *root* password, ask the administrator who installed your system.

To log in as the super user, follow these steps:

1. When you see the login message, enter the super user login name:

   ```
   login: root
   ```

   Now press the ⟨Return⟩ key. The system prompts you for the super user password.

2. Enter the super user password and press the ⟨Return⟩ key. The system does not display the password as you enter it, so enter each keystroke carefully.

The system opens the super user account and displays the message of the day and the super user prompt "#".

Take special care when you are logged in as the super user. In particular, you should be careful when deleting or modifying files or directories. This is important because the super user has unlimited access to all files; it is possible to remove or modify a file that is vital to the system. Avoid using wildcard designators in filenames and keep track of your current working directory.

You can leave the super user account at any time by pressing ⟨Ctrl⟩**d**.

Stopping the system

Stopping a UNIX system requires more than just turning off the computer. You must prepare the system for stopping by using either the **shutdown**(ADM) or (under certain conditions) the **haltsys**(ADM) command.

Using the wall command

Before stopping the system with the **shutdown**(ADM) command, you should notify users of the impending shutdown. You may want to include other details, such as when the system will be restarted.

To send a system-wide message to the terminals of all the users who are currently logged in, use the **wall**(ADM) ("write to all") command:

wall

Press ⟨Return⟩. Enter the message, pressing ⟨Return⟩ to start a new line if necessary. When you have finished entering the message, press ⟨Ctrl⟩**d**. This displays your message on all system terminals.

Using the shutdown command

The **shutdown** command is the normal way to stop the system and should be used whenever the system is in normal operation mode. It warns other users that the system is about to be stopped and gives them an opportunity to finish their work. (The warning message that **shutdown** displays at all terminals can be customized; see the **shutdown**(ADM) manual page for details.) If desired, the system administrator can also use the **wall**(ADM) command to send a message about the impending shutdown prior to running the actual **shutdown** command.

To stop the system with the **shutdown**(ADM) command, follow these steps:

1. Log in as the super user. Refer to "Logging in as the super user" (page 51) for more information. The system opens the super user account and displays the message of the day and the super user prompt.
2. Enter the following command and press ⟨Return⟩:

   /etc/shutdown -g*n*

   Δ **sysadmsh** users select: System ➪ Terminate

   where ***n*** is the number of minutes before the shutdown is to take place. The system displays a warning message at each terminal, asking logged-in users to finish their work and to log out. As soon as all users are logged out or the specified time has elapsed, the system closes all accounts and displays the following message:

```
   ** Safe to Power Off **
            -or-
** Press Any Key to Reboot **
```

3. Turn off the computer or press any key to reboot the system.

Using the haltsys command

The **haltsys**(ADM) command halts the system immediately. This command should be used only when in single user mode. If there are any users logged into the system when the **haltsys** command is given, they are logged out and their work in progress is lost. In addition, network servers and other programs are terminated abnormally and could create problems when they are restarted.

To stop the system with the **haltsys** command, follow these steps:

1. You should be in single-user mode (use **who -r** to check). Remember that the **haltsys** command should not be used in multiuser mode.

2. Enter:

   /etc/haltsys

   Now press the ⟨Return⟩ key. The system displays the following message:

```
   ** Safe to Power Off **
          -or-
** Press Any Key to Reboot **
```

3. Turn off the computer, or press any key to reboot the system.

Understanding the boot display information

At boot time, a table of hardware information is always displayed after the copyright information. This table represents your hardware configuration as recognized by the operating system. Example 3-1 is an annotated version of the boot screen as it appears on a sample machine.

Example 3-1 Sample boot display

```
device    address     vector dma comment
--------------------------------------------------------------------------
fpu       -             35   -  type=80387
floppy    0x03F2-0x03F7 06   2  unit=0 type=96ds15
serial    0x02F8-0x02FF 03   -  unit=1 type=Standard nports=1
parallel  0x0378-0x037A 07   -  unit=0
console   -             -    -  unit=ega type=0 12 screens=68k
disk      0x01F0-0x01F7 36   -  type=W0 unit=0 cyls=791 hds=16 secs=48
```

This information is explained in the following key:

device, address, vector, dma, comment	the name of the hardware, address in hexadecimal, interrupt vector, direct memory access channel, other details about the hardware, respectively
fpu	floating-point unit present, specifically the Intel 80387 and 80486 chips
floppy	high density 5.25" floppy drive (type=96ds15)
serial	this is COM1; COM1 has one port (nports=1, no multiport card is installed)
parallel	this is parallel port lp0 (unit=0)

(Continued on next page)

(Continued)

console	the console has an EGA video adaptor (unit=ega) compatible with (type 0) the IBM EGA design
disk	Western Digital st506 controller number 0 (W0), hard drive 0 (unit 0), as well as the number of cylinders, heads, and sectors

The **hwconfig**(C) utility displays or accesses this information at any time, using the configuration information stored in the file */usr/adm/hwconfig*. Refer to the **hwconfig**(C) manual page for more information.

In addition, the **eisa**(ADM) utility can be used to list the cards installed in EISA machines, and the **slot**(ADM) utility used for MCA machines.

Changing the boot process

Each time the computer is started, the system runs the **boot** program. Unless you give different instructions at the prompt, **boot** loads the default kernel program using the configuration values specified in the file */etc/default/boot* on the default root filesystem. You can change the current boot process with your response to the prompt, or you can edit the */etc/default/boot* file to change the default configuration values for future boot operations. You can also set certain options in */etc/default/boot* to allow the system to boot automatically. The **boot**(HW) manual page describes the default boot options.

Booting an old kernel

If you have recently relinked the kernel and it fails to boot properly or causes other problems, you can enter **unix.old** at the boot prompt and load the previous kernel.

Changing the default bootstring (DEFBOOTSR)

To change which program is loaded by default when you enter only a ⟨Return⟩ at the boot prompt, modify the default bootstring set with the **DEFBOOTSTR** option in */etc/default/boot*. For example, the following setting in */etc/default/boot* causes the **boot** program to load the kernel from a hard disk by default.

```
DEFBOOTSTR=hd(40)unix
```

The first two letters of the argument specify the device ("hd" for hard disk or "fd" for floppy disk). The number in parentheses is the minor device number (40 for the root filesystem on the hard disk). Following the parentheses is the pathname of the program to be loaded.

The *Hardware Configuration Guide* and the **boot**(HW) manual page describe certain keywords or "bootstrings" that you can add to the **boot** command line to load special drivers at boot time. You can also add these to the end of the default bootstring set with **DEFBOOTSTR**. For example, here is the bootstring for a Wangtek cartridge tape:

```
DEFBOOTSTR=hd(40)unix ct=wangtek(0x338,5,1)
```

Booting automatically

The settings of the **AUTOBOOT**, **TIMEOUT**, and **PANICBOOT** options in */etc/default/boot* control if, and when, automatic booting occurs.

If **AUTOBOOT=NO**, the **boot** program waits indefinitely for a response to the prompt. If **AUTOBOOT=YES**, the system boots automatically if no response is given at the prompt after a certain amount of time. The amount of time **boot** waits for a response before booting automatically is 60 seconds by default, but you can set the number of seconds to wait with the **TIMEOUT** option. If a timeout occurs, **boot** behaves as though you entered a ⟨Return⟩ to the boot prompt, performing the default boot process using the configuration values specified in */etc/default/boot*.

You can set the **PANICBOOT** option to **YES** or **NO** to indicate whether or not the system should reboot after a panic (a panic always causes the system to halt). Refer to the **autoboot**(ADM) manual page for more information.

Chapter 4

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 protect them from access by other users. Each account has a unique "login name" and "password" with which the user enters the system, and a "home directory" where the user works. In addition, the system has certain defaults that define how long a user password should last, whether users are allowed to choose their own passwords, and how many unsuccessful login attempts should be allowed before locking a user out.

It is the system administrator's job to create accounts for all users on the system, and to maintain these accounts by changing user passwords, login groups, and other account parameters when necessary.

There is the option to have several system administrators, each charged with limited control over certain aspects of system operation, or a single *root* user controlling all administration operations (*root* exists with full super user powers by default).

This chapter covers the following topics:

- Account information: for those who are familiar with UNIX systems, this section explains how database files such as */etc/passwd* and */etc/shadow* are used and how they coexist with the additional C2-related databases.
- Account management: how to add, alter, and remove (or retire) user accounts, plus create user groups.
- Default account configuration: how to configure system default login and password parameters, and if desired, change the security scheme.

It is important to examine the default account restrictions soon after creating user accounts. Refer to "Default account configuration" (page 81) for a summary. You should determine if these defaults are appropriate to the needs of your system.

How account information is stored

One of the principal differences between the various UNIX systems is in how account information is stored. This affects the migration of accounts across different UNIX systems, and governs how programs access this data. The 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. The following files are supported and maintained by the system to ensure compatibility with other UNIX systems (files marked with an asterisk are TCB files):

- */etc/passwd* file. 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. It can be edited by experienced administrators, but **sysadmsh** is the preferred method for adding and maintaining user accounts. If your system is configured with the Improved or High security defaults and you edit */etc/passwd* manually, you must run the **authck**(ADM) command with the **-p** and **-y** options to update the Protected Password database. In the lower modes, you can edit */etc/passwd* without running **authck**.
- */etc/shadow* file. This file is readable only by *root*. It contains the encrypted password otherwise found in the */etc/passwd* file. (When the */etc/shadow* exists, */etc/passwd* contains an "x" in the password field.) This file is manipulated via the **pwconv**(ADM) and **pwunconv**(ADM) utilities. This file exists by default in all security defaults except Low, where it still can be created using **pwconv**.
- */etc/default/passwd* and */etc/default/login* files. These contain default account information that, in many cases, is duplicated in the Protected Password and System Defaults database.

* 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 as well as their privileges (authorizations), password parameters, and other detailed information. The format of this file is described in the **authcap**(F) manual page.

* System Defaults database (*/etc/auth/system/default*) file. This contains the information used for creating default accounts; the contents of this file are determined by the security defaults selected (Low, Traditional, Improved, or High). The format of this file is described in the **authcap**(F) manual page.

All database files are updated automatically when a change is made via **sysadmsh**.

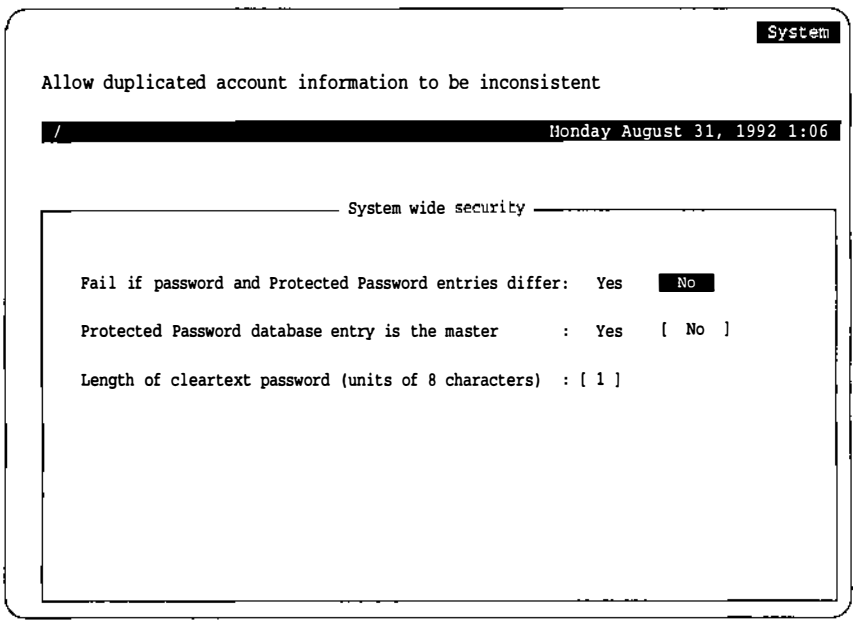
In the event of a discrepancy between these files, either the UNIX System V files or the TCB databases are used as the master to bring them into agreement. In the Low and Traditional security defaults the UNIX System V files are the master*. You can also configure which set of files is used as the master set; this is described in the next section.

Configuring database recovery and precedence

To define the recovery scheme to be used when a database failure occurs, make the following **sysadmsh** selection:

Accounts ➪ Defaults ➪ System

The following screen is displayed:



* Refer to "Default account configuration" (page 81) for information on security defaults.

The following parameters define database redundancy and precedence:

Fail if password and Protected Password entries differ
: If Yes, this enforces consistency in account information; an error message is generated when an inconsistency occurs and no action is taken. If No, inconsistent data is permitted (and automatically corrected) between the different database files. Under the Improved and High security defaults, users are locked out of the system when a discrepancy occurs until the administrator logs in on the override terminal to fix the problem.

Protected Password database entry is the master
: If Yes, this ensures that the trusted database files are used as the master files and information is written to the traditional UNIX system data files only for consistency — it is never relied upon for data used by the system. If No, the traditional System V data files (*/etc/passwd* and so forth) are used as the master files and the trusted databases (Protected Password and System Default) are written to only for consistency and are not relied upon for data used by the system. Under the Low and Traditional security defaults, the System V files are used as the master.

Length of cleartext password
: Because the password encryption schemes on most other XENIX and UNIX systems support a maximum password length of eight characters (pre-encryption), this field provides compatibility by controlling the maximum length of passwords. The term "cleartext" refers to an unencrypted password, so password length in this instance is expressed in units of eight characters. Under the Low and Traditional security defaults, this field has a value of 1, which maintains compatibility with other UNIX systems (meaning that the */etc/passwd* file can be imported to other systems).

Account management

This section explains how to create and manage user accounts.

Adding a user

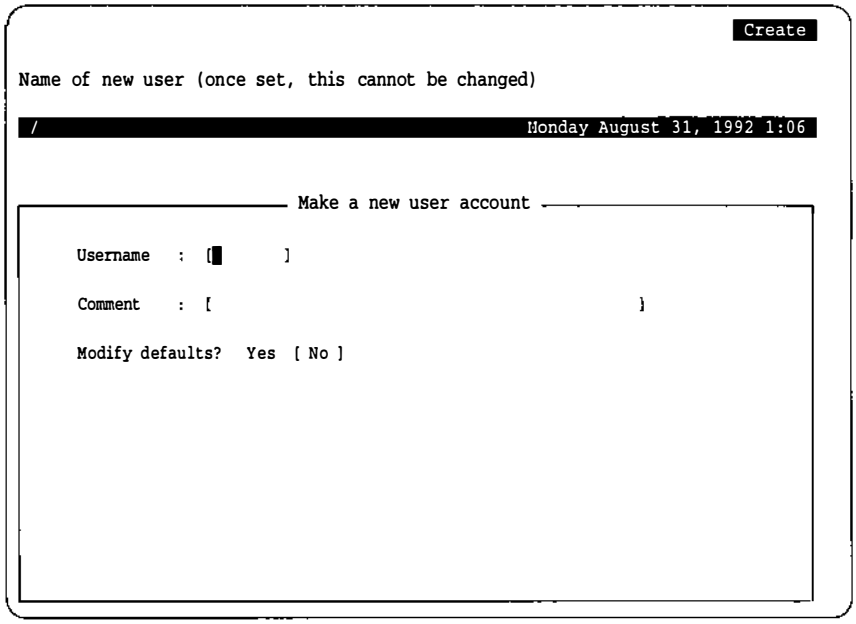
You can add a user account to the system with **sysadmsh**(ADM), which creates a new entry in the Accounts database. The database contains information about the new user (such as login name and initial password) that the system uses to let the user log in and begin work. **sysadmsh** also creates a home directory for the user, a mailbox for use with the **mail** command, and an initialization file (for example, *.profile* for the Bourne and Korn shells or *.login* for C-shell) containing UNIX system commands that are executed when the user logs in.

NOTE Although **sysadmsh** is the preferred method for adding and maintaining user accounts, experienced administrators can edit the */etc/passwd* file directly. If your system is configured with the Improved or High security defaults and you edit */etc/passwd* manually, you must run the **authck**(ADM) command with the **-p** and **-y** options to update the Protected Password database. In the lower modes, you can edit */etc/passwd* without running **authck**.

To create a user account, make the following **sysadmsh** selection:

Accounts ⇨ User ⇨ Create

The following screen is displayed:



Follow these steps to add a user:

1. Fill in the "Username" and, if desired, "Comment" fields.
2. If you wish to alter the defaults, select Yes and define the fields as shown in "Modifying account creation defaults" (page 64) . Fill in each field as necessary; press ⟨F3⟩ to choose from point-and-pick lists. When you press ⟨Return⟩, the field is filled in with the value you selected.
3. When you exit the form, a window pops up to confirm your additions. If confirmed, a series of creation messages are displayed that look like this:

```
Created home directory: pathname
Created shell file: filename
Greetings mail sent to user: name
```

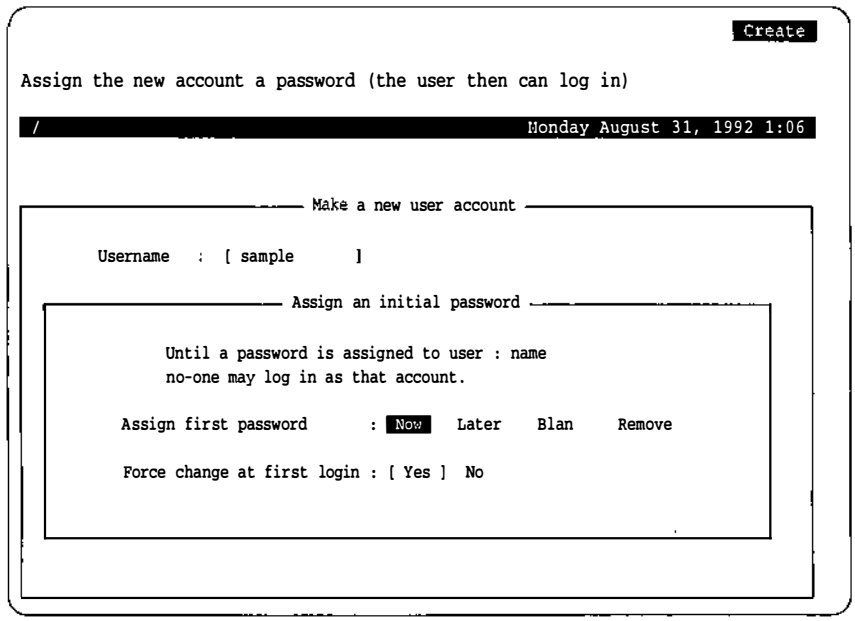
This indicates that all the necessary files and directories were created. (This default information is taken from the directory */usr/lib/mkuser*.)

NOTE If you see a message similar to the following:

```
useshell: Warning
useshell: File Control database inconsistency
useshell: Script path /usr/lib/mkuser/mkuser.init may be compromised
          - run integrity(ADM)
```

this means there may be a problem with the permissions of the database file indicated (including any part of the path). If your system is configured with the Low or Traditional security defaults, this message can be ignored. For Improved or High defaults, this can be remedied by using the **fixmog**(ADM) utility to correct permissions on the system (use **fixmog -i** to fix any problems interactively). We recommend that you run the **integrity**(ADM) utility first to get a report of all incorrect permissions on the system, but note that **integrity** does not make changes.

4. Finally, **sysadmsh** prompts you as to whether an initial password should be created. The following menu is displayed:



The possible selections for "Assign first password" are as follows:

Now	assigns the new account a password
Later	does not assign the new account a password (the user cannot log in)

Blank — assigns the new account an empty password (the user is asked to set a password at first login)

Remove — assigns the new account no password at all (the user can log in without a password)

The "Force change at first login" field governs whether users must change their passwords the first time they log in. Note that setting this to Yes negates the effect of Remove; that is, the user is required to set a password when logging in.

5. If you selected to generate a password for the new user, you see the following prompt:

```
Setting password for user user
Password change is forced for user

                    Choose password

You can choose whether you pick your own password,
or have the system create one for you.

          1. Pick your own password
          2. Pronounceable password will be generated for you

Enter choice (default is 1):
```

6. If you select 1 and you are operating under the Improved or High security defaults, you see the following prompts:

```
Please enter new password (at least 5 characters):
Please choose a password which contains a mixture of lower-
and upper-case letters, digits (0 - 9), and non-alphanumeric
characters (e.g., !, #, @, ;, %, or /.)

Please do NOT choose a password that is an English word,
or which is the name of a person, place, or thing, or which
contains the string "SCO", "XENIX", or "UNIX" (in either case).
Re-enter password:
```

Note that the password is not displayed on the screen as you enter it. You are asked to enter another password if the one you entered is unacceptable.

7. If you select 1 and you are operating under the Low or Traditional security defaults, you see the following prompts:

```
Please enter new password (at least 1 character):

New password:
```

You can also simply press ⟨Return⟩ to establish a null password, where the user presses ⟨Return⟩ when prompted for the password.

8. If you select 2, the following is displayed:

```
Generating random pronounceable password for user.
The password, along with the hyphenated version, is shown.
Hit <RETURN> or <ENTER> until you like the choice.
When you have chosen the password you want, type it in.
Note: Type your interrupt character or 'quit' to abort at any time.

Password:xxxxxxxx  Hyphenation:xx-xx-xx Enter password:
```

   The generated password is displayed with a hyphenated version. The hyphenation separates the password into pronounceable syllables and is designed to help you commit the password to memory.

9. Give the new password to the user. If you selected to force a password change, the user is required to change it immediately after logging in for the first time.

The new account is usable and is maintained according to the default security parameters unless you have set specific values for the user.

Modifying account creation defaults

For most users, the defaults should prove sufficient. If you selected to "Modify Defaults," you see the following form:

```
                                                                    Create

Use the system default login group

/                                                  Monday August 31, 1992 1:06

                          Make a new user account
                          New user account parameters

     Login group    : Specify Default of
                      Value, (F3) for list :
     Groups         : [...            ]

     Login shell    : Specify [Default] of sh
                      Value, (F3) for list :

     Home directory : Specify [Default] of [ /usr/name                  ]
                      Value, (F3) for list : [                              ]
                      [ Create home ]    Do not create     Populate existing

     User ID number : [Specify] [Default] of [     ]  value : [     ]

     Type of user  : Specify [Default] of individual
                      Value, (F3) for list :
     Account that may su(C) to this user   :
```

The cursor is initially positioned on the "Login group" field. Some of the fields displayed can only be modified at creation time, in the Modify mode. These fields are informational only and their values cannot be changed.

The fields are as follows:

Login group
: This is the group associated with this account when the user logs in. This field can be changed, but must not be empty. It becomes the group field for this user in */etc/passwd*. Pressing function key ⟨F3⟩ provides a point-and-pick list of all currently existing groups.

Groups
: These are the groups this user is a member of. If you enter the name of a group that does not exist, it is automatically created as described in "Adding or changing groups" (page 76).

Login shell
: This is the shell the user will use. (The default is defined in */etc/default/authsh*.) If a full pathname is supplied (as in */bin/sh*), the shell described by that pathname is simply used as the user's login shell. However, if the shell specified is not a pathname (as in *sh*), it is assumed to be the name of a "predefined shell," a shell defined in a subdirectory of */usr/lib/mkuser*. Choosing a predefined shell causes appropriate shell-related files (for example, *.profile* for *sh*) to be copied into the user's home directory when the account is created. (Refer to "Adding login shells and configuration files" (page 94) for more information on creating your predefined shells.)

Home directory
: This defines where the user's files will reside. The default directory option is highlighted. Press ⟨Return⟩ to get the default location. The home directory options are as follows:

Create	creates a new directory for this user
Do not create	does not create a directory for this user
Populate existing	uses the existing directory specified

If you wish to have users share a single home directory, refer to "Sharing home directories" (page 66) for instructions.

User ID number
: This is the user identification (ID) number. Once set, a user's identity should not be changed as this would violate accountability.

Type of user
: By default this is "individual," and you need not change it. Pseudo-users are anonymous accounts like *sysinfo* and *uucp*. 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.) Several user types are provided for customers who need to make distinctions between accounts that can be used by different people.

Account that may su(C) to this user

This is the user responsible for this account. This field is only useful with the High security defaults, where strict accountability is maintained and use of **su**(C) is restricted. Under other defaults, this field can be ignored. This field can be changed if and only if the user is not an individual. For individual users this field is empty, but for non-individuals it must not be empty. It must contain the user name of another account. For example, all pseudo-user accounts shipped with the system are owned by *root*, and defined as an "individual" account. This can be set up so that every account can be traced to a real person. Press ⟨F3⟩ for a point-and-pick list of all users on the system.

Sharing home directories

You can set up user accounts to share the same login directory. To do this, create the directory normally during the adding of a new user. You should then exit **sysadmsh** and enter the following commands (replacing ***homedir*** with the real directory name):

cd ***homedir***
chmod 775 .
chown auth .

In addition, enter one of the following commands according to the login shell used for the account:

Bourne or Korn shell:

chmod 660 .profile
chmod 660 .kshrc (Korn shell only)

C-shell:

chmod 660 .login .cshrc

This ensures that the members of the same login group can share this directory. Note that if you assign a different login group to a user, that user cannot share the directory. To add users that share the directory, be sure to select "Modify defaults," specify the directory, and "Populate existing" directory when creating subsequent users. Answer **n** when asked to overwrite the "dot" files such as *.profile* and so forth.

Adding administrative users

In addition to the standard Identity information, users who act as administrators for printers, accounts, and so forth, can be assigned the responsibility by selecting Privileges. Under lower security defaults, most subsystem authorizations are already assigned by default, so this distinction is meaningless. (The **auth** authorization gives *root* powers and is only assigned to *root* by default, regardless of the security defaults.)

WARNING The **auth** subsystem authorization should only be assigned to persons entrusted with account administration. This is because it permits the user to make changes to any account, including *root*.

Subsystems are discussed in "Changing default authorizations" (page 90) and assigning authorizations is discussed in "Changing user authorizations" (page 74).

Altering the defaults displayed for user accounts

You can alter the default selections that appear in the user creation menu by editing the file */etc/default/authsh*. The following defaults can be redefined:

- login group
- groups
- login shell
- home directory
- range of user IDs
- type of user

The fields in */etc/default/authsh* are explained in the **authsh**(ADM) manual page.

Removing or retiring a user account

When operating under High (C2) security defaults, a user is never removed from the system. Once assigned, a user ID (UID) is never reused. Instead, a user account is "retired," or removed from service. Under other defaults, a user can be completely removed from the system. To retire a user account, make the following **sysadmsh** selection:

Accounts ⇨ User ⇨ Retire

When running with lower security defaults, you can completely remove accounts using the **rmuser**(ADM) utility. To remove a user, enter the following command, substituting the name of the user account for ***username***:

rmuser ***username***

Retiring or removing a user account does not remove the user's files; the system administrator must do this manually.

Unretiring a user account

If you are running under lower security defaults and you wish to "unretire" or reactivate a user's account, you can do so with the **unretire**(ADM) command. To reactivate a retired user, enter the following command, substituting the name of the user account for ***username***:

unretire ***username***

Locking or unlocking a user account

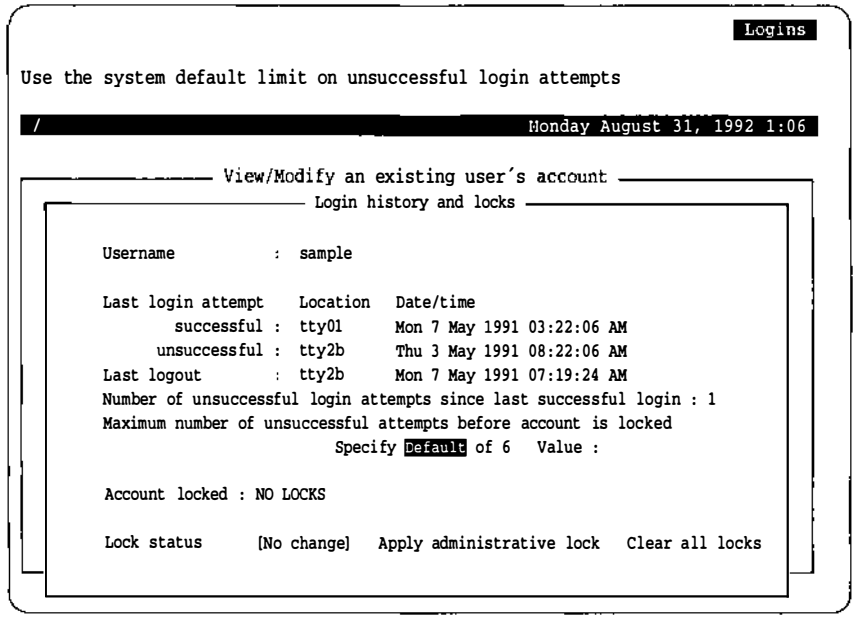
The system administrator can lock an account to prevent its use. In addition, under the Improved and High security defaults, an account is locked automatically if certain login parameters have been exceeded (see "Default account configuration" (page 81) for more information).

NOTE Under the Low and Traditional security defaults, accounts and terminals are only locked by the administrator; there are almost no limits on the number of unsuccessful login attempts, and password expiration is less strict.

Once a user or terminal is locked, only an administrator can unlock the user account or terminal (terminal locks are discussed in the next section). To lock or unlock an account, make the following **sysadmsh** selection:

Accounts ➪ User ➪ Examine:Logins

The colon indicates that you must fill in a field (in this case the user name) before choosing the Logins selection. A form similar to the following is displayed:



Move down to the "Lock status" field and toggle it to "Apply administrative lock" or "Clear all locks" as desired.

Locking or unlocking a terminal

To lock and unlock a terminal, respectively, use the following **sysadmsh** selections:

Accounts ➪ Terminals ➪ Lock
Accounts ➪ Terminals ➪ Unlock

When the prompt appears for the terminal, enter the name, for example: tty01. When a terminal is locked, the following message is displayed when an attempt is made to log in:

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

Changing a user's login group

To change a user's login group, make the following **sysadmsh** selection:

Accounts ➪ User ➪ Examine:Identity

The colon indicates that you must fill in a field (in this case the user name) before making the Identity selection. A form similar to the following is displayed:

```
                                                                    Identity

Group associated with this account when the user logs in (<F3> for list)

/                                                  Monday August 31, 1992 1:06

                    View/Modify an existing user's account
                                  Identity

     Username          :  sample

     User ID           :  246                  Type of user :   individual
                          Account that may su(C) to this user :   NONE

     Login group       :  [ pub          ]
     Groups            :  [ ...          ]

     Login shell       :  [ /bin/sh                                        ]
     Home directory    :  [ Keep ]   Edit    Create    Move    Restore
              path     :    /usr/sample
     Comment           :  [                                                ]
     Priority          :  Specify [Default]  of  0  Value:
```

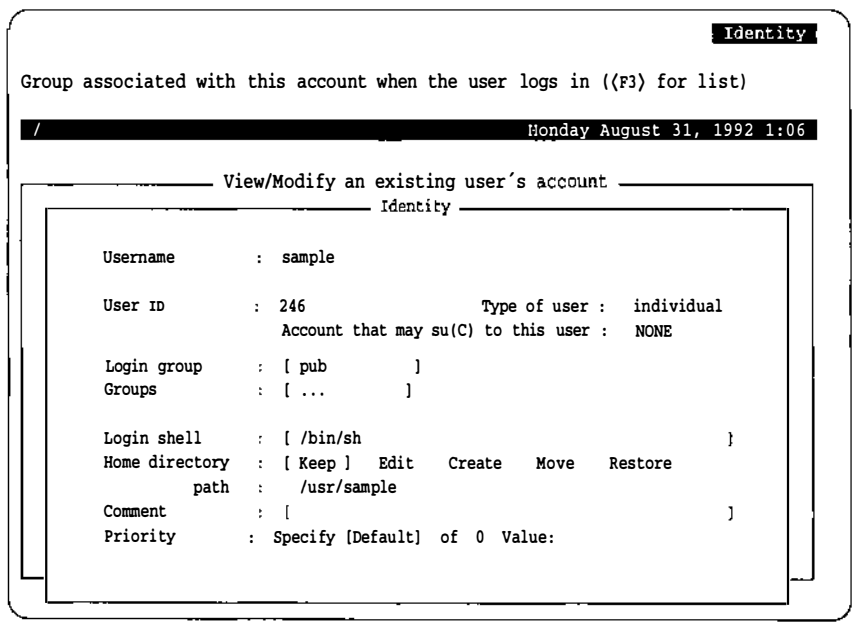
Modify the "Login group" field as desired.

Changing a home directory

To change a user's home directory, including moving the user's files, make the following **sysadmsh** selection:

Accounts ⇨ User ⇨ Examine:Identity

The colon indicates that you must fill in a field (in this case the user name) before making the Identity selection. A form similar to the following is displayed:



After moving down to the "Home directory" field, you can select to do the following:

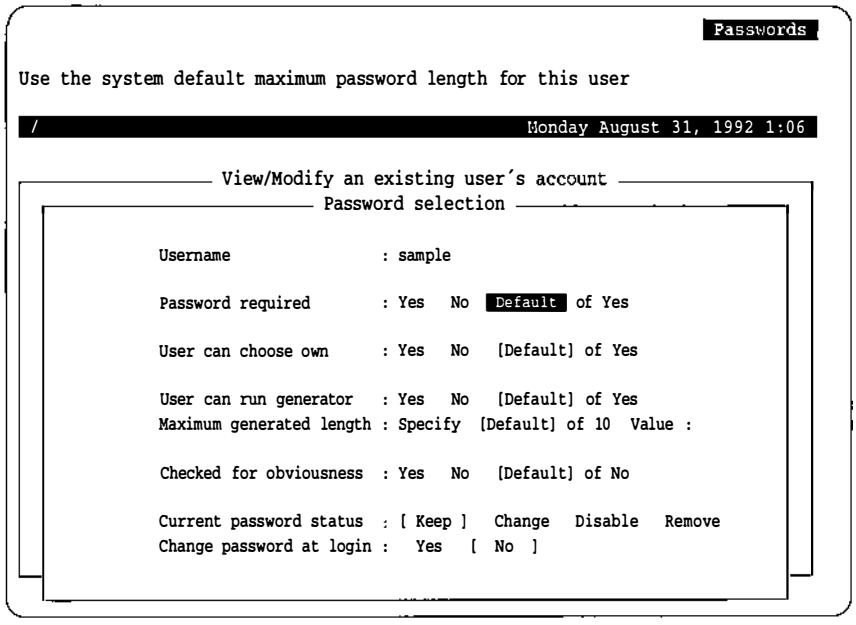
Keep	makes no change to the main directory for this user
Edit	changes this user's main directory path, but does not move any files
Create	creates the main directory for this user
Move	renames this user's main directory, moving all files in the old to the new
Restore	changes the path back to its previous value (no files are moved)

Changing a user password or password parameters

An administrator can change a user's password at any time. Password generation parameters can also be changed on an individual basis, just as they can be system-wide. This governs how a user's password is changed. To do this, make the following **sysadmsh** selection:

Accounts ⇨ User ⇨ Examine:Password

The colon indicates that you must fill in a field (in this case the user name) before making the Password selection. The following form is displayed:



The Password selection parameters are similar to the system-wide parameters described in "Changing default password restrictions" (page 87). The following parameters define the per-user password restrictions:

Password required
: If Yes, the user cannot log in without a password; if No, the user can log in without a password.

User can choose own
: This parameter determines whether or not users can choose their own passwords. If this parameter is set to Yes, users can pick their passwords. If that parameter is set to No, the system must generate a password for that user, according to the random password generation procedures.

> **NOTE** If "User can choose own" and "User can run generator" are both No, users cannot change their own passwords.

User can run generator
: This parameter enables the user to run the password generator. Note that this does not allow the user to choose a password, but merely to generate a new random password.

Maximum generated password length
: This parameter is the maximum length of a password generated by the system for this user. The maximum is 80 characters.

Checked for obviousness
: This parameter is only valid with Improved or High security defaults, or when **GOODPW=YES** appears in */etc/default/passwd*. This controls whether the system should run simple or complex triviality checks on a new password. Triviality checks increase the time required to change a password substantially. Refer to "Customizing password checking with goodpw(ADM)" (page 89) for more information.

Current password status
: You can choose one of the following:

  Keep — does not change password.

  Change — invokes the password change procedure described in "Adding a user" (page 60).

  Disable — disables the password, which effectively locks the user out.

  Remove — removes the password, enabling the user to log in without a password.

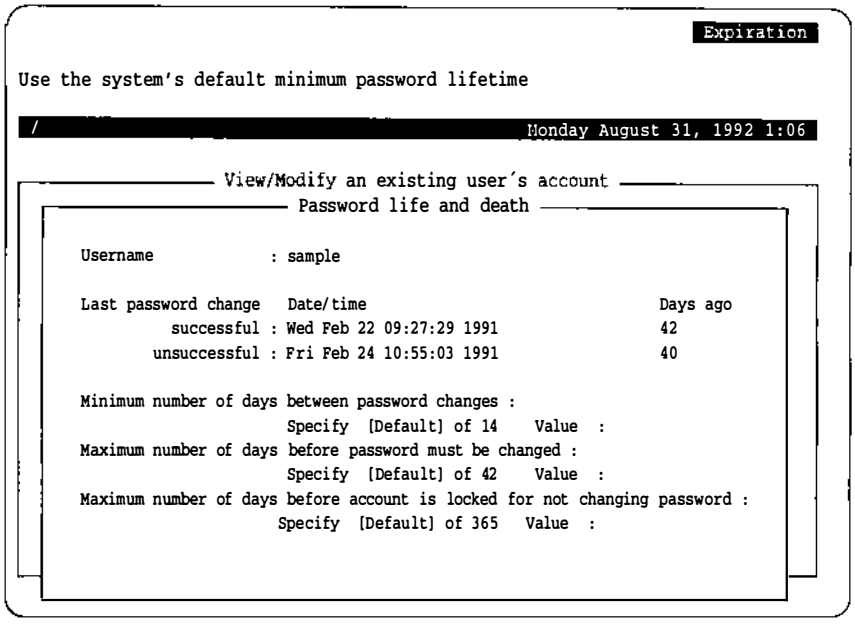
Change password at login
: This option allows you to force users to change their password the next time they log in.

Altering user password expiration parameters

It is sometimes useful to define expiration parameters for a user that differ from the system defaults. To do this, make the following **sysadmsh** selection:

Accounts ⇨ User ⇨ Examine:Expiration

The colon indicates that you must fill in a field (in this case the user name) before choosing the Expiration selection. A form similar to the following is displayed:



The user parameter descriptions are similar to the system-wide parameters described in "Changing default password restrictions" (page 87). The descriptions differ, but the parameters are the same. They control the time that a password is valid, and the procedures for changing the password once it becomes invalid. The lifetime of a password is divided into three intervals:

- The password is valid.
- The password has expired; the user can still log in and change it (if authorized to do so).
- The password is dead; the user is locked out and the administrator must unlock the account and the user change the password.

NOTE The Low and Traditional security defaults 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 system also stores a minimum time between password changes. This prevents users from changing their passwords when they expire and then immediately changing them back to one they remember. A user's password cannot be changed until the minimum time has been exceeded. The password lifetime scheme is implemented as follows:

Minimum number of days between password changes
: This defines the number of days users must wait before they can change their password.

Maximum number of days before password must be changed
: This defines the length of time a given password is valid.

Maximum number of days before account is locked for not ...
: This defines the interval between the last password change and when the password dies.

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.

Changing user authorizations

Authorizations define what users are allowed to do on the system. Authorizations are discussed in detail in "Changing default authorizations" (page 90). Authorizations are also used to assign administrative roles; for more information see "Administering subsystems with sysadmsh" (page 207).

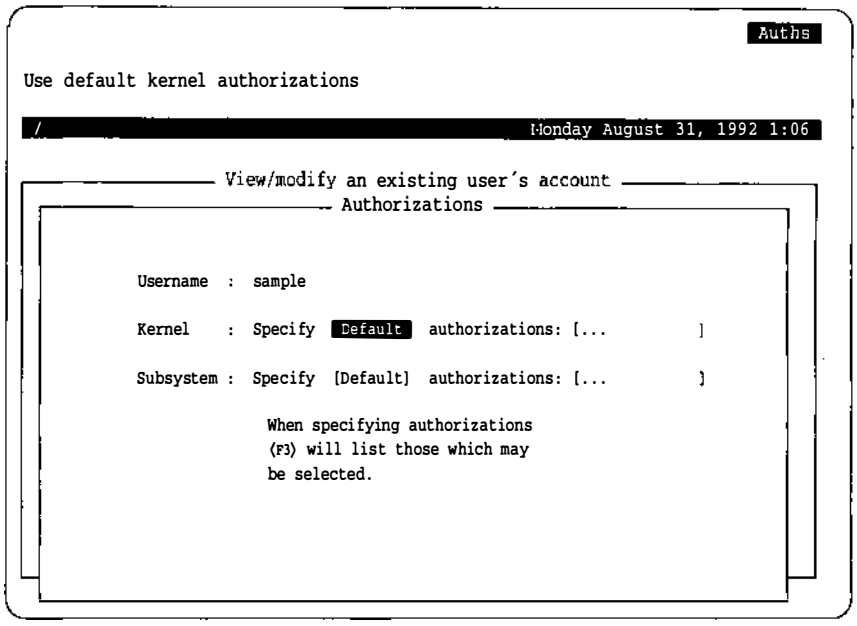
NOTE The Low and Traditional security defaults assign most authorizations to users by default and it should not be necessary to change them. With the High (C2) security defaults, few authorizations are assigned by default; they are intended for users entrusted with administration of a subsystem. Only users entrusted with system administration should be assigned the **auth** subsystem authorization.

To assign a new authorization to a user, make the following **sysadmsh** selection:

Accounts ➪ User ➪ Examine:Privileges

The colon indicates that you must fill in a field (in this case the user name) before choosing the Privileges selection.

The following form is displayed:



You can select Specify and press ⟨F3⟩ to open a window that lists the available authorizations. Use the arrow keys to move up and down the list. You can select multiple authorizations by pressing the space bar; each item selected is marked with an asterisk (*).

NOTE If you switch from defaults to specified, the default values are eliminated for that user; only those authorizations you specify are in effect.

Changing user audit parameters

You can define audit parameters for individual users just as with the system-wide parameters. Any settings defined for a user override the system defaults.

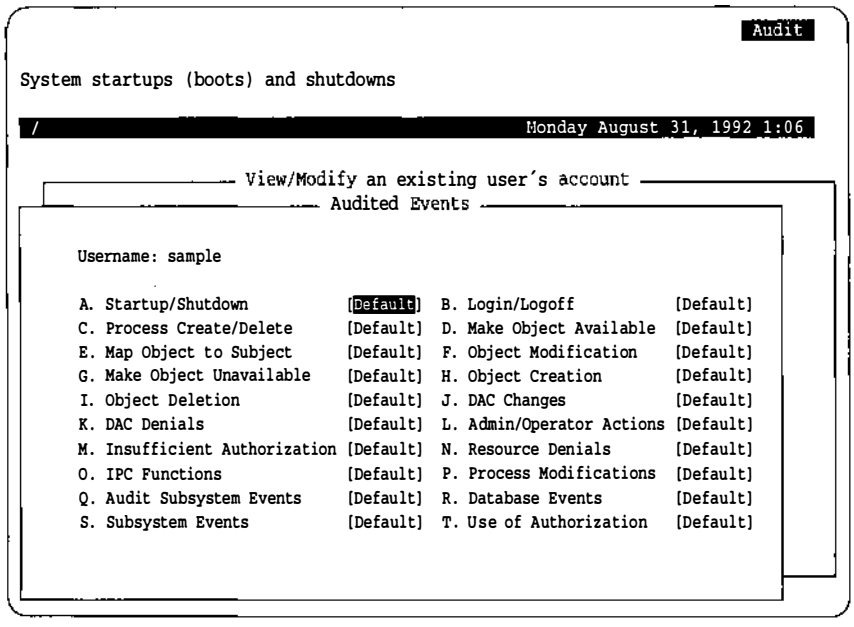
NOTE It is not necessary to modify audit parameters if you do not intend to use auditing.

To define or change audit parameters, make the following **sysadmsh** selection:

Accounts ➪ User ➪ Examine:Audit

The colon indicates that you must fill in a field (in this case the user name) before choosing the Audit selection.

A form similar to the following is displayed:



Refer to "Audit event types" (page 244) for a detailed description of audit events.

There are three possible settings for each event:

Default	use the system-wide account defaults
Always	always audit this event (overrides system-wide default)
Never	never audit this event (overrides system-wide default)

You can press ⟨F3⟩ to select from a list of these settings or fill them in manually. Abbreviations are recognized (for example, "n", "nev", and "N" all mean Never). To execute the form, press ⟨Ctrl⟩x. (If you fill in the last field on a form, it is automatically executed.)

Adding or changing groups

To add a group, simply enter a new group name while creating or altering a user account. You are prompted that the group does not exist and asked to confirm that you wish to create the new group.

You can also edit the file */etc/group* and add or modify the contents as desired.

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

Supplemental groups

On most System V implementations, a user can be a member of several groups, but can only have a single effective group ID (GID), which limits membership to one group at a time. The effective GID (and group membership) is changed using the **newgrp**(C) command. The list of groups is located in */etc/group*.

The supplemental groups feature allows an additional set of groups to be defined, permitting simultaneous membership in multiple groups. This provides additional access permissions on top of those provided by the user and group IDs.

The *.suppgroups* file, which is recognized by programs that establish identity (**login**, **su** and **cron**), specifies which groups (of which the user is a member) are placed in the user's supplemental group list when identity is established. The *.suppgroups* file contains one group name per line. If a *.suppgroups* file does not exist, the initial supplemental group list is filled with the login group ID followed by the groups the user is listed as being a member of in */etc/group*. They are listed in the same order as in */etc/group*. The *.suppgroups* file does not need to be readable or writable by the user.

The **sg**(C) command is used to manipulate a user's supplemental group set. This command does not establish identity so it does not read the *.suppgroups* file.

Changing the maximum number of supplemental groups

By default, the maximum size of a supplemental group set is eight, meaning that a user can only be in eight groups at one time. This number is controlled by the **NGROUPS** tunable kernel parameter. This value can be changed by invoking the following **sysadmsh** selection:

System ➪ Configure ➪ Kernel ➪ Parameters

and selecting category 3, "Files, Inodes, and Filesystems," and changing the value of **NGROUPS**. The kernel must then be relinked and booted for the new value to take effect. You can use this selection to relink the kernel:

System ➪ Configure ➪ Kernel ➪ Rebuild

See "Reallocating kernel resources with configure" in the "Tuning system performance" chapter of this guide for complete instructions.

Allowing users to execute super user 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**. The procedure for setting up a super-user command for this usage is described in detail in the **asroot**(ADM) manual page.

NOTE The **asroot** utility requires the user's password to be re-entered when the system is configured with the High security defaults.

Accessing other accounts with su(C)

The **su**(C) utility (for super user) 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 the following 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). To exit the shell, enter **exit** or press ⟨Ctrl⟩**d** and you are returned to your own account.

Users can **su** to an account they are defined as responsible for in the "Account that may su(C) to this user" field of the Accounts ➪ User ➪ Examine:Identity **sysadmsh** selection. To access the *root* account (or any other account they are **not** responsible for), however, the user must have the **su** authorization. This can be assigned using the Accounts ➪ User ➪ Examine:Privileges selection. Refer to "Changing user authorizations" (page 74) for more information.

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

Moving user accounts to another filesystem or directory

If you want to move your user accounts to a different location (such as a new filesystem or directory), you must start by ensuring that new accounts are placed in the new location. Here we use a hypothetical filesystem called */x* as an example. The **sysadmsh**(ADM) Accounts option (used to create new user accounts) reads the default location for user accounts from the */etc/default/authsh* and */usr/lib/mkuser/homepaths* files.

Edit the files */etc/default/authsh* and */usr/lib/mkuser/homepaths*. Change the entries that read "/usr" to "/x". This establishes */x* as the location for new user accounts.

Whenever you run the **sysadmsh** Accounts selection to add a new user, that user account is placed in */x*. (Make certain that the filesystem containing the user accounts is mounted before you create a new user, or the user's home directory will not be accessible when the filesystem is mounted.)

If there are existing user accounts that you wish to relocate, you should first make a backup floppy or tape of all home directories you plan to move. (This protects your users from possible loss of data.) Next, change the home directory for each user by calling up their account information with the following **sysadmsh**(ADM) selection:

Accounts ➪ User ➪ Examine:Identity

Move to the "Home Directory" field and select the Move option. Follow the prompts. Do this for every user whose home directory has changed.

Migrating user accounts to non-SCO-based 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*, you should 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-based UNIX system, use the **ap**(ADM) utility described in the next section.

Password compatibility

The password encryption scheme used by the 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 defaults, passwords of up to 80 characters are allowed, which cannot be imported to other systems. The **sysadmsh** selection Accounts ➪ Defaults ➪ System includes the field "Length of cleartext password". A value of 1 (instead of 10 as 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. Refer to "Configuring database recovery and precedence" (page 59) for more information.

Migrating user accounts to SCO-based UNIX systems

You can easily duplicate user accounts on other SCO-based UNIX systems (as with a network) using the **ap**(ADM) utility. This utility creates a profile containing all account data for one or more users.

WARNING The **ap**(ADM) utility does not create profiles that are portable to non-SCO-based UNIX systems. If you need to migrate accounts from a non-SCO-based UNIX system or SCO XENIX system, use the **addxusers**(ADM) utility described in the next section.

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, do the following:

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

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

   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 the following command:

   ap -r -f profile.acct *usernames*

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

Migrating user accounts from SCO XENIX or non-SCO-based UNIX systems

To copy accounts from other non-SCO-based UNIX systems (or SCO XENIX systems) to your system, use the **addxusers**(ADM) utility. **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 more information.

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

Default account configuration

This section explains how to alter the system security defaults, which include default password schemes, subsystem authorizations, and number of login attempts permitted for users.

The system includes four sets of defaults that define the security scheme for user accounts:

- Low
- Traditional
- Improved
- High

The "High" 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*). Table 4-1 lists the defaults used by each set. The Low and Traditional defaults use values consistent with non-trusted UNIX systems. Selecting the security defaults is described later.

NOTE The pre-defined defaults can be customized as desired; the four default sets are provided for ease of configurability.

Table 4-1 System default security parameters

Security parameters	Security defaults			
	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
Checked for obviousness	no	UNIX	yes[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)	0	1	2	2
Time to complete login (secs)	60	60	60	60
Authorizations				
Subsystem	audittrail, backup, lp, mem, queryspace, shutdown, su, terminal	audittrail, mem, printqueue, queryspace, terminal, su	audittrail, queryspace, printqueue, su	queryspace
Kernel	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	yes	yes	yes
SUID/SGIDclear 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]	UNIX	UNIX	TCB	TCB
Other				
Users can schedule jobs	allow	allow	deny	deny
Home directory mode	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 disallowing user names, machine names, etc...
2. Thorough checks are made, including disallowing words that appear in the on-line 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 235).
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 defaults, 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 recovery and precedence" (page 59).
8. This feature allows the suppression of login messages. See **login**(M) for information.

Changing the security defaults

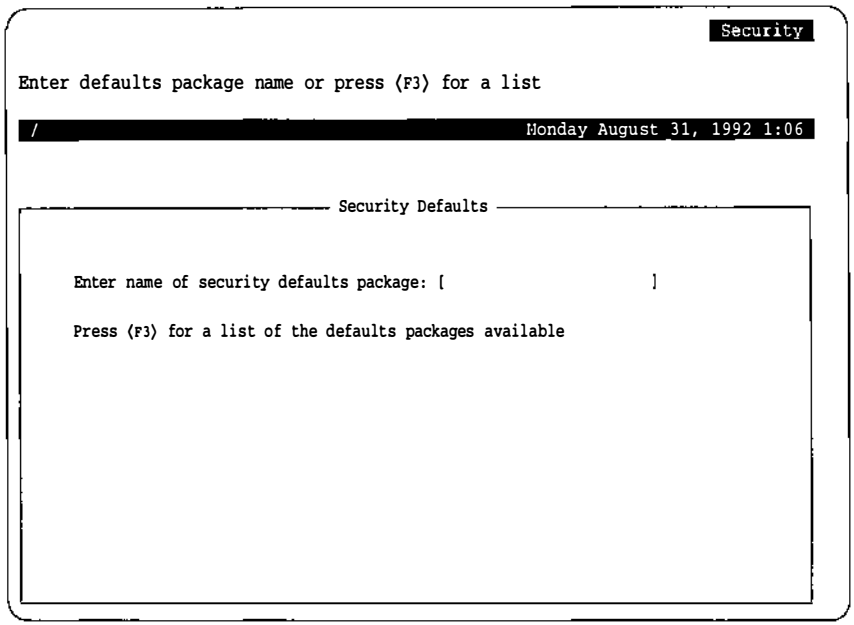
You were given the choice of security defaults at installation time. It is possible to later select another set of defaults. Should you wish to change the security defaults of your system, make the following **sysadmsh** selection:

System ➪ Configure ➪ Security

This allows you to select an alternate set of defaults (see Table 4-1).

NOTE After having selected lower security defaults, it is possible to select the Improved or High defaults, although this does not mean that your system automatically conforms to the requirements of a C2 system. (By definition, a C2 system must adhere to the requirements from initial installation.)

When you make the Security selection, the following is displayed:



You may reconfigure your system security to suit your own requirements. Four sets of preconfigured security defaults are available:

High — recommended for systems containing confidential information and accessed by many users. Passwords are strictly controlled and assigned to users; they cannot choose their own. User accounts cannot be removed or reactivated. All C2 features are engaged and 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. Database corruption results in system lockout.

Traditional — provided for compatibility with existing 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 not publicly accessible with 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 system is designed to meet the requirements for the C2 level of trust describing the protection given to prevent unauthorized access to a system and its data. The High and Improved defaults are suitable for C2 systems.

Changing system account parameters dynamically

In addition to selecting a package of default parameters, the following system-wide account parameters can be customized individually:

- authorizations
- password
- logins

The system-wide security parameters control the way that users log in and, once they establish a session, the terminal and authorization environment that the system presents to them. Each parameter that you can change from the **sysadmsh** interface is discussed here. Other parameters that affect system operation are addressed later.

You should use the system-wide functions to define your own default system behavior. Then use the user-specific functions to adjust that behavior for any user having different requirements. As you might expect, the user-specific entries override the system defaults for any given user.

Changing default login restrictions

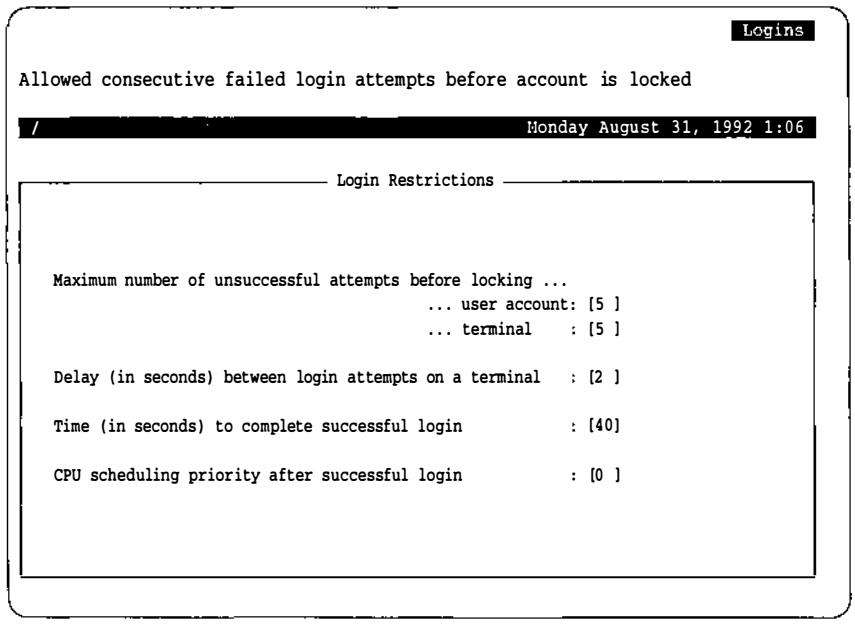
Most of the configurable security parameters deal with the way the system creates a login session. These include login particulars and the way that passwords are generated and enforced. The login parameters enforce the account and terminal-locking features. When users log in, they must give a login name and password. In addition, the user has a limited number of tries to log in. There is a limit on the number of times an unsuccessful login attempt can occur before either the account or the terminal are locked. If either count is exceeded, the user or the terminal is locked against future login. This feature guards against penetration attempts by restricting the number of times a malicious user (or computer programmed by a malicious user) can try to break into the system.

NOTE The Low and Traditional security defaults use login restrictions that are very lenient; there are almost no limits on the number of unsuccessful login attempts and little delay between attempts.

To access the login restriction parameters, make the following **sysadmsh** selection:

Accounts ➪ Defaults ➪ Logins

The following form is displayed:



The parameters are described as follows:

Maximum number of unsuccessful attempts before locking
: This is the system default number of unsuccessful attempts allowed for users and terminals. If a particular user or terminal needs either a more restrictive or more permissive number, the user's account can be modified or the terminal's configuration can be changed to override the system default. Refer to "Terminal login management" (page 217) for more information.

Delay (in seconds) between login attempts on a terminal
: This parameter controls the amount of time that must pass between unsuccessful login attempts. To further reduce the possibility of penetration, the system can delay between login attempts to increase the amount of time it takes to try to log into the system repeatedly. You can increase this parameter to control the cycle time of the "login:" prompt. By combining this parameter with the user and terminal unsuccessful attempt parameters, you can frustrate attempts to try passwords repeatedly on certain (or a combination of) terminal lines.

Time (in seconds) to complete successful login
: This parameter determines how much time users have to enter their name and password before the login attempt is terminated.

CPU scheduling priority after successful login

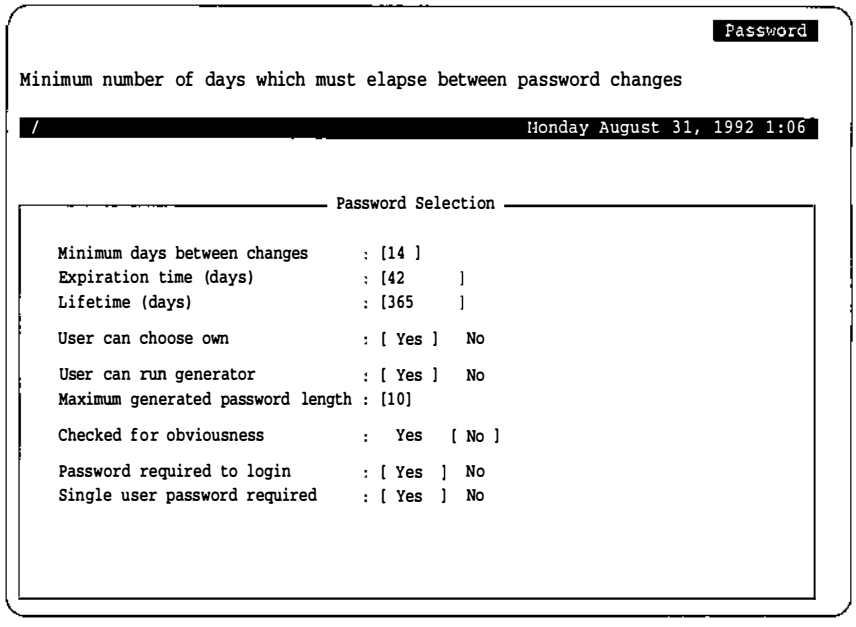
This sets the **nice**(C) value associated with this user's processes. This allows you to set a higher or lower CPU priority for a user. See the **nice**(C) manual page for details.

Changing default password restrictions

Given that you can control the number of attempts an intruder can try to guess a password, the remaining task is to control the complexity of the password itself. To access the password restriction parameters, make the following **sysadmsh** selection:

Accounts ⇨ Defaults ⇨ Password

The following screen is displayed:



The types of password checking the system does is controlled by the parameters set on this screen. The parameters control the time that a password is valid, and the procedures for changing the password once it becomes invalid. A password is valid until it "expires" or "dies." An expired password can be changed by whomever is authorized to change passwords for the account. A password expires when its expiration time is reached. The expiration time can be set from **sysadmsh** on a system-wide or a per-user basis, and it is expressed in number of days from the time that the password was last changed. A dead password causes the user account to be locked. Only the administrator can unlock the user's account, which is then treated as an account with an expired password. The password must still be changed before the user can log in again.

NOTE The Low and Traditional defaults use password restrictions that are very lenient; passwords do not expire, accounts are not locked, and there is no minimum interval between password changes. In addition, there is no password checking with the Low defaults; Traditional defaults use the customary UNIX system password checks.

The system also stores a minimum time between password changes. This prevents users from changing their passwords when they expire and then immediately changing them back to one they remember. A user's password cannot be changed until the minimum time has been exceeded. This parameter may also be set on a per-user or system-wide basis.

The following parameters define the password restrictions:

Minimum days between password changes
: The number of days a user must wait between password changes.

Expiration time (days)
: This defines the length of time a given password is valid. 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.

Lifetime (days)
: This defines the interval between last password change and when the password dies.

User can choose own
: This parameter determines whether or not users can choose their own passwords. 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. Most UNIX systems allow users to pick their passwords. If this parameter is set to Yes, then rules consistent with non-trusted UNIX systems are in effect, allowing users to pick their passwords. If that parameter is set to No, the system must generate passwords for that user, according to the random password generator.

User can run generator
: This parameter enables users to run the password generator. Note that this does not allow users to choose a password, merely generate a new random password.

Maximum generated password length
: This defines the maximum length of a password generated by the system. The maximum is 80 characters.

Checked for obviousness
: This parameter is only valid with the Improved or High security defaults, or when "GOODPW=YES" appears in */etc/default/passwd*. This controls whether the system should run simple or complex triviality checks on passwords. These checks assure that the password does not appear in the online dictionary, along with the other checks described in **goodpw**(ADM). Setting this parameter to Yes ensures the failure of some penetrations based on dictionary checking, but this can be controlled more effectively through the limits on account and terminal logins. Triviality checks increase the time required to change a password substantially. Refer to "Customizing password checking with goodpw(ADM)" (this page) for more information.

Password required to login
: If No, user accounts can be without passwords; if Yes, a password must exist for the account.

Single user password required
: This governs whether a password is required to bring the system up in single-user (maintenance) mode.

When an account is locked by the system, only *root* or the accounts administrator can unlock it. The password must be changed at that time. Refer to "Changing a user password or password parameters" (page 71) for information on overriding these parameters for any user.

Customizing password checking with goodpw(ADM)

The **goodpw**(ADM) utility 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, user, group, and system names.

NOTE Password checking can also be set by editing the */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

In addition, the directory */usr/lib/goodpw/checks* allows you to further customize password requirements according to the type of password:

- user
- filsys (filesystem)
- group
- modem

Each type has a file (*secure* or *weak*) that is used depending on whether the "Check for obviousness" field is set to Yes (*secure*) or No (*weak*).

You can also define regular expressions (character combinations and arrangements) that all passwords must match, or must not match, with the files */usr/lib/goodpw/match* and */usr/lib/goodpw/reject*, respectively. For more details, refer to the **goodpw**(ADM) manual page.

Changing default authorizations

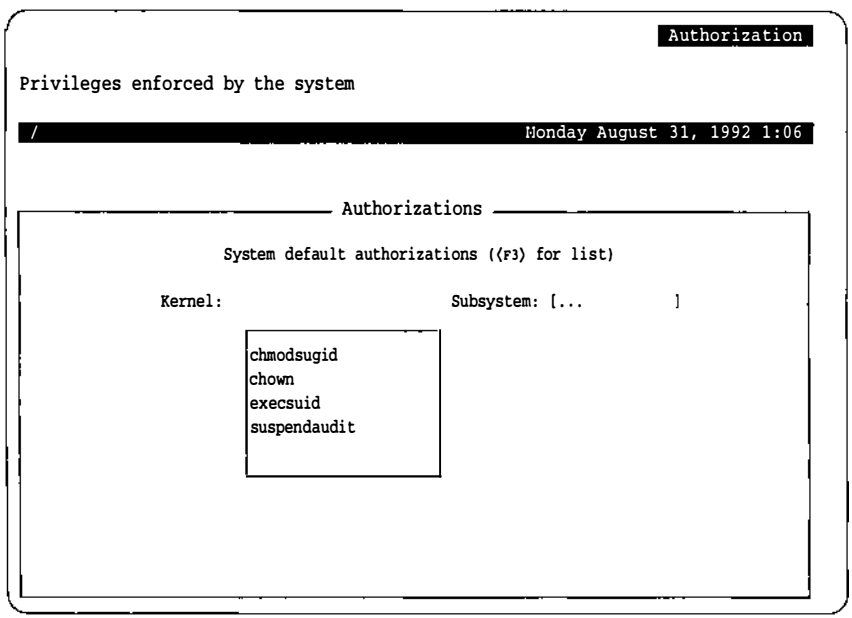
There are two types of authorizations: *Subsystem authorizations* are associated with users and allow the user to execute trusted utilities. *Kernel authorizations* are associated with processes and allow a process to perform certain actions. Each user session has a set of kernel authorizations and subsystem authorizations.

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*. Refer to the sections that follow for more information.

To access the authorization parameters, make the following **sysadmsh** selection:

Accounts ➪ Defaults ➪ Authorizations

The following screen is displayed:



You can use the ⟨Tab⟩ key to move between kernel and subsystem authorizations. Use the ⟨F3⟩ key to get a pop-up window that lists each set of authorizations. Detailed descriptions of authorizations appear in the sections that follow.

Subsystem authorizations

Subsystem authorizations were designed to implement administrative roles rather than using a single *root* user to administer the system. Under the Low and Traditional security defaults, most subsystem authorizations (except **auth**) are assigned to users by default.

If you intend to operate a system that conforms to C2 requirements, you should grant subsystem authorizations based on the notion of "least-privilege", where users are assigned 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. Secondary authorizations are provided to grant limited capabilities from a subsystem.

The subsystem authorizations are listed in Table 4-2.

Table 4-2 Subsystem authorizations

Authorization	Subsystem	Powers
mem	memory	access to system data tables, listing all processes on the system
terminal	terminal	unrestricted use of the **write**(C) command
lp	line printer	administer printers
backup	backups	perform backups
auth	accounts	administer accounts: adding users, changing passwords, and so on
audit	audit	audit administrator: run system audits and generate reports
cron	job scheduling	control use of **cron**(C), **at**(C), and **batch**(C) commands
root	asroot(ADM)	allows use of any command found in */tcb/files/rootcmds* (see "Allowing users to execute super user commands" (page 77) for more information)
sysadmin	–	not implemented
passwd	passwords	ability to change user passwords

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 defaults to provide behavior that is more consistent with other UNIX operating systems. See Table 4-3.

Table 4-3 Secondary authorizations

Secondary authorization	Subsystem	Description
audittrail	audit	ability to generate personal audit reports on one's own activities
queryspace	backup	use of **df**(C) command to query disk space
printqueue	lp	view all jobs in queue using **lpstat**(C)
printerstat	lp	use of printer enable/disable commands
su	auth	access to the *root* (super user) account and other accounts. (Access still requires password.)
shutdown	asroot(ADM)	ability to use the **shutdown**(ADM) command.

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.)

Kernel authorizations

The kernel authorizations govern the power that user processes have to execute specific operating system services. For example, the ability to change ownership of a file is governed by the **chown** authorization. (The **chown** authorization allows the use of the **chown**(S) system call which enables **chown**(C) to work.) The default kernel authorizations are used whenever a user's kernel authorizations are unspecified. Thus, users that need more authorization can have user-specific entries that grant them those authorizations, while normal users can have their authorizations set to the system-wide defaults. See Table 4-4.

NOTE Restricted **chown** is required for NIST FIPS 151-1 conformance. The **chown** authorization should not be assigned to users if you wish to conform to NIST FIPS 151-1 requirements.

Table 4-4 Kernel authorizations

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

Under the Low and Traditional security defaults, most kernel authorizations are assigned by default and should not require modification. Under the High security defaults, **chmodsugid** is not assigned by default. Most users require only **execsuid** kernel authorizations to perform routine tasks. If the user needs to create files with the SUID or SGID bits, they must have the **chmodsugid** authorization. To change ownership of a file (give it away), the **chown** authorization is required. If a user does not have this authorization, ownership of files can only be changed by *root*. The audit kernel authorizations (**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.

Kernel authorizations and administrative users

If you are operating with the High and Improved security defaults, you must assign certain kernel authorizations along with subsystem authorizations. Although most of these are already assigned by default, they are listed in Table 4-5 in case you modify the defaults. One exception is the audit subsystem, which requires the addition of the **configaudit** and **suspendaudit** authorizations. These authorizations should never be assigned by default, or to ordinary users.

NOTE Under the Low and Traditional security defaults, most kernel authorizations are already assigned by default.

Table 4-5 Subsystem kernel authorization requirements

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

Adding login shells and configuration files

It is possible to add files for additional login shells that can be selected for users when they are created using **sysadmsh**. 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*. You can examine the existing files and follow their example. Make sure the permissions and ownership are consistent with the other files. When you install the files, the new shell is a valid selection in the "Login shell" field of the user Identity form.

Chapter 5

Managing filesystems

This chapter describes one of the most important responsibilities of a system administrator: creating and maintaining filesystems. General maintenance activities are described, such as strategies for maintaining free space. The concept of "filesystem integrity" is introduced, with a description of how the operating system repairs damaged filesystems. Filesystems reside on hard disks, CD-ROMs, and floppy disks. Adding hard disks and CD-ROM drives is described in the *Hardware Configuration Guide*. After adding a new device to the system, filesystems residing on them must then be added to the system using **mkdev fs** as described in this chapter. For information on file permissions and other security considerations, see the chapter entitled "Maintaining system security."

What is a filesystem?

A filesystem is a distinct division of the operating system, consisting of files, directories, and the information needed to locate and access them. A filesystem can be thought of as a structure upon which directories and files are constructed.

Each UNIX system has at least one filesystem on the primary hard disk. This filesystem is called the "root filesystem" and is represented by the symbol " / ". The root filesystem contains the programs and directories that comprise the operating system. On small hard disks, the root filesystem also includes all the user directories. The primary hard disk can also be divided into more than one filesystem. This is described in the *Installation Guide*. One of the most common divisions is the */u* filesystem, used to isolate user accounts from the root filesystem.

A UNIX system can also have other filesystems that contain special directories and application programs. Dividing the primary hard disk into multiple filesystems protects the data and makes maintenance easier. Adding still more filesystems by installing other hard disks expands the system storage space. New filesystems can be specifically created by the system administrator, then "attached" (mounted) and "detached" (unmounted) when needed, in the same way that a floppy disk is accessed. The process of adding a hard disk is described in the *Hardware Configuration Guide*. Adding filesystems after a hard disk is configured is described in the following section.

Adding new filesystem(s)

After configuring a new hard disk, you must then add the new filesystem(s) to the system. To do this, follow these steps:

1. Enter the following command:

   mkdev fs

   Δ **sysadmsh** users select: Filesystems ⇨ Add

2. You see the following:

```
This program performs maintenance tasks required to add or delete
an existing filesystem. Would you like to:

        1. Add a new filesystem to system.
        2. Remove a filesystem.

Select an option or enter q to quit:
```

   Select 1.

3. You are next prompted for the device name:

```
Enter a device name and press <Return> or q to quit:
```

   Enter the full pathname of the device from */dev*. The device name is derived from the name you gave the filesystem during the **divvy** phase. For example, to add a filesystem called *u*, you enter "/dev/u".

4. You are now prompted to provide the name of the mount point to be used:

```
Enter a directory name and press <Return> or q to quit:
```

   This directory is where the filesystem is mounted. For example, a filesystem called *u* is mounted on the directory */u*.

5. The following is displayed:

```
Reserving slots in lost+found directory ...

When entering multiuser mode:
        1. Always mount filesystem
        2. Never mount filesystem
        3. Prompt before mounting filesystem

Select an option:
```

   If you want the filesystem mounted automatically at system startup, enter 1. If you wish it mounted only by the request of the system administrator, select 2. If you select 3, the system prompts you at system startup whether or not you want the filesystem mounted.

6. You are then asked whether or not you want to permit users to mount filesystems:

```
Do you want to allow users to mount this file system? (y/n)
```

   You must respond "y" so that the system backup program can mount and unmount the filesystem as necessary.

7. The following messages are displayed when the process is complete.

```
Updating system files ...
Filesystem has been successfully added.
```

8. Next, you should mount the */x* filesystem using the following command:

   mount /dev/x /x

   Δ **sysadmsh** users select: Filesystems ⇨ Mount

Mounting and unmounting a filesystem

The **mount**(ADM) command attaches a filesystem. For example, to mount or unmount */dev/u* on */u*, you would use the following two commands, respectively:

mount /dev/u /u

Δ **sysadmsh** users select: Filesystems ⇨ Mount

umount /u

Δ **sysadmsh** users select: Filesystems ⇨ Unmount

Only the super user can use the **mount** and **umount** commands.

NOTE Files in a filesystem are not accessible unless the filesystem is mounted. If files are copied to or created in the mount point directory while the filesystem is unmounted, those files will appear to be in that filesystem when they are not. When the filesystem is mounted, these files seem to "disappear" when the filesystem is mounted over them.

Permitting users to mount filesystems

Only the super user can use the **mount** command. However, the super user can set up parameters to define which filesystems can be mounted by users with the **mnt**(C) command. These parameters may include an access password, if desired.

Each filesystem must have an entry in the file */etc/default/filesys*. Example 5-1 contains a sample set of entries.

Example 5-1 Sample /etc/default/filesys file

```
bdev=/dev/root  cdev=/dev/rroot  mountdir=/  \
desc="The Root Filesystem"  rcmount=no  mount=no

bdev=/dev/u  cdev=/dev/ru  mountdir=/u  rcmount=yes  \
fsckflags=-y  desc="The User Filesystem"

bdev=/dev/x  cdev=/dev/rx  mountdir=/x  mount=yes  \
rcmount=yes  fsckflags=-y  desc="The Extra Filesystem"
```

The sample entries determine the behavior shown in Table 5-1.

Table 5-1 Filesystem mount specifications

Filesystem	When mounted	Can user mount?
root	boot time	no
/u	multiuser	no
/x	anytime	yes

If you wish to make any non-root filesystem mountable by users, simply add "mount=yes" to the entry for the given filesystem. In addition, when the **mnt** command is invoked without an argument (no filesystem name), the program checks all non-root filesystems to see if they can be mounted and, if so, mounts them. The option "mount=prompt" asks the user if they want to mount each filesystem where a mount is permitted.

For more information on the **mnt** command, including a complete list of options, refer to the **mnt**(C) manual page.

Filesystem types

Your system can be set up with five different types of filesystem:

- XENIX
- S51K (UNIX filesystem)
- DOS
- AFS (Acer Fast Filesystem)
- EAFS (Extended Acer Fast Filesystem)

The default filesystem is the EAFS (Extended Acer Fast Filesystem), which supports long filenames (filenames exceeding 14 characters) and symbolic links (file links across filesystems). The block sizes for each of these filesystems are 1K. DOS filesystems are discussed in "Mounting DOS filesystems on a UNIX system" (page 196).

The XENIX, S51K, and AFS filesystems are provided for backwards compatibility. The S51K and AFS filesystems can be easily converted to EAFS as described in this section.

Converting AFS filesystems to EAFS

An AFS filesystem can be converted to an EAFS filesystem. This is done using the **-E** option of the **fsck**(ADM) command, which normally checks and repairs filesystems. The **-E** option also changes the format of the filesystem to EAFS.

NOTE You must unmount a filesystem before running **fsck**.

The command has the form:

fsck -E ***device***

where ***device*** is the name of the filesystem device in */dev*.

Converting UNIX filesystems to AFS

A UNIX system filesystem can be converted at any time to an AFS filesystem. This is done using the **-C*clustersize*** option of the **fsck**(ADM) command, which normally checks and repairs filesystems. The **-C** option alters the clustersize to change the format of the filesystem to AFS. The ***clustersize*** argument must be a power of 2 and less than 16 (8 is the recommended value).

The real benefits of the AFS filesystem are seen with a new filesystem. The increase in speed that is possible with a converted AFS filesystem is not immediately apparent; it takes effect only as new files are added to the filesystem. There is little or no benefit in transforming a filesystem that is nearly full; if it is within a few blocks of being full, the conversion does not work. (Refer to "Filesystem integrity" (page 114) for more information.)

NOTE You must unmount a filesystem before running **fsck**.

The command has the form:

fsck -s -C*clustersize device*

where ***device*** is the name of the filesystem device in */dev*. Note that the **-s** option must also be present.

Configurable filesystem features

There are certain filesystem features that are configurable, including:

- Group ID of newly created files: you can specify whether a new file has the group affiliation of the parent directory or that of the creating user.
- Filename truncation: you can specify what happens when an attempt is made to create a filename longer than the system limit.

This section explains these features.

Setting directory SGID bit

By default, the GID (group identifier) of a newly created file is set to the GID of the creating process or user. This behavior can be changed by setting the SGID bit on a directory. Setting the SGID bit on a directory results in a new file having the GID of that directory. To set the SGID bit on a directory, enter the following command, where ***directory*** is the directory name:

chmod g+s ***directory***

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

Setting filename truncation

By default, attempts to create filenames longer than the system limit result in the error message "Filename too long." This can be changed so that long filenames are silently truncated to the system limit. The system limit depends on the filesystem type. XENIX, S51K, and AFS filesystems have a limit of 14 characters. The EAFS (default) filesystem has a limit of 255 characters. The default behavior is mandated by POSIX FIPS requirements and is controlled by the **ETRUNC** kernel parameter. This parameter can be changed by invoking

the **sysadmsh** selection System ⇨ Configure ⇨ Kernel ⇨ Parameters and selecting category 3: "Files, Inodes and Filesystems," and changing the value of **ETRUNC** to 1. The kernel must then be relinked and booted for the new behavior to take effect. Use the **sysadmsh** selection System ⇨ Configure ⇨ Kernel ⇨ Rebuild to relink the kernel. See "configure(ADM) — reallocate kernel resources" (page 504) of this guide for complete instructions.

Using links

A link is a directory entry referring to a file. The same file can have several links. This allows a given file to appear wherever it is required without the need for separate files. Any changes made to the file are effectively independent of the name by which the file is known. This means that no matter which link is modified, the same file is sourced. Hard links cannot be made across filesystems and cannot refer to a directory (see the "Symbolic links" section).

The syntax for a hard link is as follows:

ln ***file target***

where ***file*** is the name of the existing file, and ***target*** is the name of the new directory entry which will source the same data. The long file listing generated by the **l**(C) command looks like this example:

```
-rw-rw-r--  2 stevem   pub  60091 Feb  7 19:54  help.file
```

The number "2" that follows the file permissions indicates the number of links. Note that there is no way to distinguish a hard link to a file from its original directory entry.

Symbolic links

Symbolic links allow you to connect files or directories on different filesystems. The **-s** option to the **ln**(C) command allows you to do this.

NOTE Symbolic links are only available on EAFS filesystems.

The syntax is as follows:

ln -s ***file target***

You can easily recognize a symbolic link using the long file listing generated by the **l**(C) command:

```
lrwxrwxrwx   1 stevem   group   13 Feb 10 15:34 text -> /u/forbin/file1
```

The arrow (->) points literally to the location of the actual file. Unlike hard links, the source file is always discernable from its symbolic links.

If ***target*** is a directory, then one or more files are linked to that directory. If ***source*** is a directory, a directory link is made.

Maintaining free space in filesystems

An important task of the system administrator is filesystem maintenance, which includes keeping the system running smoothly, keeping the filesystems clean, and ensuring adequate space for all users. To maintain the filesystems, the system administrator must monitor the free space in each filesystem, and take corrective action whenever the free space gets too low.

This section explains the filesystem maintenance commands. These commands report how much space is used, locate seldom-used files, and remove or repair damaged files.

A UNIX system operates best when at least 15% of the space in each filesystem is free. In any system, the amount of free space depends on the size of the disk containing the filesystem and the number of files on the disk. Because every disk has a fixed amount of space, it is important to control the number of files stored on the disk.

If a filesystem has less than 15% free space, system operation usually becomes sluggish. If no free space is available, the system stops any attempts to write to the filesystem. This means that the user's normal work on the computer (creating new files and expanding existing ones) stops.

The only remedy for a filesystem that has less than 15% free space is to delete one or more files from the filesystem. The following sections describe strategies for keeping free space available.

Strategies for maintaining free space

The system administrator should regularly check the amount of free space on all mounted filesystems and remind users to keep their directories free of unused files. You can remind users by including a reminder in the */etc/motd* (message of the day) file.

In addition, the **cleantmp**(ADM) command is run by the system to clean the */tmp* directory. You can edit the file */etc/default/cleantmp* to define which, and how often, key directories (*/tmp* and */usr/tmp* by default) are cleaned of files. See the **cleantmp**(ADM) man page for details.

If the amount of free space slips below 15%, the system administrator should follow these steps:

1. Send a system-wide message asking users to remove unused files.
2. Locate exceptionally large directories and files, and send mail to the owners asking them to remove unnecessary files.
3. Locate and remove temporary files and files named *core*.
4. Clear the contents of system log files.
5. Reduce disk fragmentation by making a complete backup of the filesystem, removing all the files, and then restoring them from the backup.
6. If the system is chronically short of free space, it may be necessary to create and mount an additional filesystem.
7. **compress**(C) large infrequently used files.

Displaying free space

You can find out how much free space exists in a particular filesystem with the **df** (for "disk free") command. This command displays the number of "blocks" available on the specific filesystem. A block is 512 characters (or bytes) of data.

The **df** command has the form:

df ***specialfile***

Δ **sysadmsh** users select: System ⇨ Report ⇨ Disk

where ***specialfile*** can be the name of a UNIX system special file corresponding to the disk drive containing the filesystem. If you do not give a special filename, then the free space of all normally mounted filesystems is given.

For example, to display the free space of the root filesystem */dev/root*, enter:

df /dev/root

The command displays the special filename and the number of free blocks.

You can find the percentage of free space to total space on your system with the command:

df -v

Sending a system-wide message

If free space is low, you can send a message to all users on the system with the **wall** (for "write to all") command. This command copies the messages you enter at your terminal to the terminals of all users currently logged in.

To send a message, enter:

wall

Enter the message, pressing ⟨Return⟩ to start a new line if necessary. After you have entered the message, press ⟨Ctrl⟩**d**. The command displays the message on all terminals in the system.

Displaying disk usage

You can display the number of blocks used within a directory by using the **du** command. This command can be used to find excessively large directories and files.

The **du**(C) command has the form:

du ***directory***

The optional ***directory*** must be the name of a directory in a mounted filesystem. If you do not give a directory name, the command displays the number of blocks in the current directory.

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

du /u/johnd

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

Use the **-a** or **-f** options to display files.

Displaying blocks by owner

You can display a list of users and the number of blocks they own by using the **quot** (for "quota") command. The command has the form:

quot ***specialfile***

The ***specialfile*** must be the name of the special file corresponding to the filesystem.

For example, to display the owners of files in the filesystem on */dev/u*, enter:

quot /dev/u

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

Mailing a message to a user

If a particular user has excessively large directories or files, you can send a personal message to the user with the **mail** command.

To begin sending a message through the mail, enter:

mail ***login-name***

where ***login-name*** is the login name of the recipient. The **mail** command copies the message to the user's mailbox, where it can be viewed by the user via the **mail** command. See the *User's Guide* for details.

Locating files

You can locate all files with a specified name, permissions setting, size, type, owner, or last access or modification date using the **find** command. This command is useful for locating seldom-used or excessively large files, or files owned by a particular user.

The **find** command has the form:

find ***pathname option***

The ***pathname*** is the pathname of the directory that you want 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***. Some options also indicate certain actions for **find** to take on located files. See the **find**(C) manual page for complete details.

Table 5-2 describes some of the options to **find** that are useful for system administrators.

Table 5-2 Useful find options

Option	Description
-atime *+number*	locates files that have not been accessed for the specified *number* of days.
-exec *cmd*	locates files that match the specified criteria and executes *cmd* on those files. The command argument ({}) is replaced by the current pathnames of the files that **find** located. An escaped semicolon (\;) must follow the *cmd* {} construction.
-group *gname*	searches for files that belong to the group *gname*. If *gname* is numeric and does not appear in */etc/group*, it is interpreted as a group ID.
-name *file*	searches for files with the specified name.
-ok *cmd*	works like **-exec** except that it displays the generated command line in prompt form; *cmd* is executed only if the user enters **y** at the prompt.
-perm *onum*	locates all files with permissions that exactly match *onum* (the octal number used with **chmod**(C)).
-print	displays the locations of any files that **find** locates.
-size *+number*	searches for files larger than the specified number of blocks (512 bytes per block).
-type *x*	locates files of a specific type; for example, type **d** for directory or **f** for a file. See **find**(C) for an explanation of the different types.
-user *uname*	locates all files that belong to the user *uname*. If *uname* is numeric and does not appear in */etc/passwd*, it is interpreted as a login ID.

NOTE If you do not include the **-print** option, **find** does not display the list of files that match the search criteria.

Finding temporary files

To locate and display all files named *temp* recursively in the */usr* directory, use the following command:

```
find /usr -name temp -print | more
```

Finding files of a certain size

You can use **find** to locate files of a specified size or type. For example, to locate and print a list of all the directories (**-type d**) greater than 3 blocks in size (**-size +3**) in all the directories (/ and below), enter:

```
find / -type d -size +3 -print
```

Finding files by permissions

Using the **-perm** *onum* option to **find**, you can locate all files with permissions that exactly match *onum* (the octal number used with **chmod**(C)). For example, to locate and display all the files in the */u* directory that give all users read, write, and execute permissions (*onum* is **0777**), enter the following command:

find /u -perm 0777 -print

Executing commands based on find output

Using the **-exec** option, you can execute a specific shell command on the files that **find** locates. The most common use of **-exec** is to locate a group of files and then remove them. For instance, when you retire a user, you can use **find** to locate all the files owned by that user, back them up, and then remove them from the system. To do this with **find**, enter:

find / -user edwarda -print | cpio -ovBc > /dev/rfd096

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

The first command locates all the files owned by user *edwarda* and copies the files to an 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.

You can instruct **find** to display a prompt (the command line that **find** generates) before executing the shell command on each file, by using the **-ok** option in place of **-exec**. When you use **-ok**, **find** prompts you with the generated command line:

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

To execute the command (in this case, **rm**), enter **y**.

The command is not executed if you enter a character other than "y", or if you press ⟨Return⟩.

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 moms "{}" \;

If you retire a user and you want to transfer ownership of their files to another user, use the following 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.

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**.

Locating core files

One useful application of **find** is to use the **-name** option to locate core and temporary files for removal.

A core file contains a copy of a terminated program. A UNIX system sometimes creates such a file when a program causes an error from which it cannot recover. 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 either core or temporary files, and they can be safely removed.

When searching for core or 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 core files in the */usr* directory that have not been accessed for one week (**-atime +7**), enter:

```
find /usr -name core -atime +7 -print
```

Once you locate the core files, you can remove them easily using one of the following two commands:

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

```
find /usr -name core -atime +7 -print | xargs rm
```

These commands find all the core files in the */usr* directory that have not been accessed in the last seven days and remove them.

Checking and clearing log files

A UNIX 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 file's previous contents. Because some log files can rapidly become quite large, it is important to check the files periodically and, if necessary, clear them by deleting their contents. It may be necessary to retain the most recently appended information in some log files, such as */etc/ddate*. In these cases, trim the files by deleting the previous data and leaving the last entry.

Some log files only grow in special situations. For example, if you want to record all attempts to use the **su** command, and you set the **SULOG** option in */etc/default/su*, the log file */usr/adm/sulog* may build up rapidly.

Table 5-3 lists some of the log files that are most likely to need clearing or trimming. Your system may have different log files from those listed, depending on its configuration and the utilities and application software installed. You may also need to check the files more or less frequently than indicated in the table, depending on system activity. Use the **find**(C) command with the **-size** flag to locate your system's unlisted large log files.

Table 5-3 Administrative log files

Logfile path	Purpose	Checking frequency
/etc/wtmp	historical login record	weekly
/usr/adm/pacct	process accounting log file	weekly
/usr/adm/messages	system messages log file	weekly
/tcb/audittmp/*	audit system temporary files	weekly
/etc/ddate	records date of each backup	monthly or yearly
/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/request	record of print requests	automatic†

† Maintenance of this file (and other files) can be accomplished automatically; see "Automatic log cleaning" (page 110).

To clear a log file and retain file permissions:

1. Copy the file to a new filename. For example:
   cp /usr/adm/messages /usr/adm/messages.old
2. You can clear the file by using one of the following 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, use the **tail**(C) command as follows:

tail -f /usr/adm/messages

With the **-f** (follow) option, **tail** prints the last ten lines of the file, followed by any lines that are appended to the file between the time **tail** is initiated and killed with the ⟨Del⟩ key.

Automatic log cleaning

It is also possible to automate the clearing of log files. using the **cron** and **crontab** facilities. By adding a line to the **crontab**(C) file for *root*, you can excute your own file maintenance scripts on a daily, weekly, or monthly basis.

The following is one of the default entries in the *root crontabs* file (*/usr/spool/cron/crontabs/root*):

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

See **crontab**(C) for complete information on the format of a *crontabs* file.

The entry for */etc/cleanup* runs each Sunday morning at 5:17. The */etc/cleanup* script is reproduced below:

```
:
#
# 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 {} \;
```

/etc/cleanup does the following once per week:

- copies the *sulog* file to *Osulog* and then clears *sulog;*
- if *filesave.log* exists, moves it 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.

Maintaining efficient filesystem organization

There are two aspects of filesystem usage that degrade the efficiency of filesystems:

- Disk fragmentation (scattering of available disk space) due to constant use and reuse of filesystem blocks.
- Directories growing too large, thus increasing search time for files.

This section explains how to avoid these problems and maintain efficient filesystem organization.

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. This increases access time and reduces system efficiency.

You can reduce disk fragmentation by first making a complete backup of all the files in the filesystem and then removing all the files from the hard disk and restoring them from the backup. Refer to "Reducing disk fragmentation" (page 567) for more information.

Because the files are completely rewritten on the disk during the restore, each file is written in one piece and fragmentation is reduced. A small amount of space is also recovered. It is a good idea to perform this action about once a year on a heavily used system and less often on a lightly used system. Be certain that you have complete, accurate, and readable backups before you begin or your files may be lost.

Large directories

It is wise not to allow directories to grow larger than necessary. You should be aware of several size limitations. A directory that contains entries for up to 62* files (plus the required "." and ".." entries) fits in a single disk block and can be searched very efficiently. A directory can have up to 638* entries and still be viable, as long as it is used only for data storage; anything larger is usually wholly unsuitable as a working directory. It is especially important to keep login directories small, preferably one block at most.

* These figures apply to filenames of 14 characters or less. As filename lengths increase, up to a maximum of 255 characters, the number of files that fit on a single disk block decreases, thus reducing the optimum number of files in a directory.

If large numbers of files are moved in and out of a directory, as in the case of a spool directory, it can also become oversized. This is because, as a rule, directories never shrink. Even if you delete files until you have less than 62 in your login directory, or less than 638 in a spool directory or other data collection directory, system searches of the directory slow down. This is because the directory slot remains the same size as it was prior to the removal of the files.

To see if a directory is too large, even if it does not seem to have over 62 files in it, use the command **hd . | wc -l**. This tells you the number of lines (inodes) in the directory listing.

The solution to the large directory problem is twofold. Educate users to keep their login directories small, and use **cpio** to back up and restore any directory that has an excess of inode numbers left over from deleted or moved files.

Limiting login directory size

To limit the size of the login directory, each user should create a hierarchy of subdirectories that reflects the use or purpose of the contained files. This can make finding files easier for the user. Emphasize the importance of setting up a simple, clear hierarchy, with only five to 10 directories at each level, so files can be added in the future without having to change the hierarchy or move files around.

Removing excess inode numbers

To remove "shadow" inode numbers (numbers left when files are deleted or moved to subdirectories), use **cpio** to back up and restore an oversized directory. Then use **hd** to list the files in the directory and see if there are any that do not show in a standard listing. These "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. Use the following procedure to remove these empty file slots from the directory. Note that the same procedure can be applied to a spool directory.

1. Move to the directory above the oversized one and create a backup directory. For example, if a user named *tracy* has an oversized home directory, enter:

   mkdir tracy.old

2. Move to the oversized directory and back it up:

   find . -print | cpio -pdlm ../tracy.old

   This command copies the directory and all subdirectories, linking files instead of copying them where possible. Confirm that the ownership, group, and permissions of the new directory and files match the original.

3. Use **l** and **hd** to check the backup directory to see that all the files are there and all the shadow inode numbers are gone.

4. Move to the directory above, then delete the oversized directory:
   rm -rf tracy
5. Rename the backup directory to replace the oversized one:
   mv tracy.old tracy

The directory appears the same to the user, but it can be searched more quickly. This can improve system performance, especially if the procedure is used on numerous working directories.

Adding disk space and filesystems

If free space is chronically low, it may be to your advantage to expand the system's storage capacity by installing additional hard disks as described in the *Hardware Configuration Guide.* Once the new disk is mounted, you can use the free space in the new filesystem for your work, or even copy user or system directories to it.

Suppose free space is low on your primary hard disk because one filesystem on it is full. If there are others with free space, or there is unused space on the hard disk, you can change the layout on the primary disk, but this is not as simple as adding a second hard disk. It is always best to plan the layout of your hard disk in advance as described in the *Installation Guide.*

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

If you decide to change the number of filesystems on your primary hard disk or to reapportion the disk space among the filesystems, you must back up your system and reinstall it as described in the *Installation Guide.* You should initialize your disk manually so that you can control the layout of your disk. During the installation process, use manual layout control to reapportion your disk space as desired.

Filesystem integrity

The most important job of the operating system is to maintain the integrity of filesystem data. Actual loss of data is a rare occurrence; UNIX system filesystems are very resistant to corruption. This is because a certain amount of redundancy exists in special structures that are invisible to the user. It is these structures that ensure filesystem integrity. For example, when the system suffers a power outage, very little information is lost. Any damage usually affects one or two files, making them inaccessible. In almost all cases, the operating system can repair any damage done to files. In very rare cases, damage causes the entire filesystem to become inaccessible.

The operating system uses the **fsck** (for "filesystem check") program to repair damaged filesystems. The **fsck** program checks the consistency of filesystems. In cases where the contents of a file are lost (rare), the only way to restore lost data is from filesystem backups. **fsck** is automatically run at boot time after an abnormal shutdown on the root filesystem. The **fsck** status messages look like this:

```
** Phase 1 - Check Blocks and Sizes
** Phase 2 - Pathnames
** Phase 3 - Connectivity
** Phase 4 - Reference Counts
** Phase 5 - Check Free List
```

If the system terminated abnormally (for example, power outage), you see other messages that may seem alarming:

```
FREE INODE COUNT WRONG IN SUPERBLK
FIX?
```

In fact, this kind of message is routine when a system was not shut down properly, and you only have to enter **y** and **fsck** continues its recovery work. This could be done without the system administrator's intervention, but it is generally better to know what is happening to a filesystem after a problem has occurred.

To discuss the idea of filesystem integrity and how **fsck** functions, it is useful to explain the basic structure of files, directories, and filesystems. Although it is not necessary to understand the principles of file storage, it is helpful to know what the messages like the one above refer to so they will seem less mysterious.

The following section describes some of the basic principles of UNIX operating systems. Refer to "Repairing filesystems with fsck" (page 117) for information on how to use the **fsck** command.

How UNIX systems maintain files

Each filesystem contains special structures that allow the operating system to access and maintain the files and data stored on the filesystem. It is the disruption and repair of these structures that we are concerned with.

The structure of a filesystem is based on the way that hard disks store data. Although the hard disk contains all the data used by the system, it is not stored in neat little locations that correspond to individual files. It is unlikely that you could point to a spot on a hard disk and truthfully say: "My file is stored right there on this part of the disk." In fact, the data is probably scattered across the disk. The operating system uses a sophisticated addressing scheme to access each of the pieces that a file is broken into and to present the data to the user as a unit.

The data is spread around because the operating system does not really deal with files, but rather with units of data. For example, assume you created a file and it is actually stored on one part of the disk. Then, suppose you edit that file and delete a few sentences here and there. This means that you are now using a little bit less disk space than when you started. This space amounts to a series of gaps in the area where your file was stored. Disk space is a precious commodity and is not wasted. So, those small amounts of disk space are allocated to other files. Picture this process on a scale of hundreds of files with a dozen users and you have an idea of how files are maintained. Because of the effectiveness of the algorithms (formulas) that the operating system uses, this process is remarkably efficient and trustworthy.

How UNIX systems maintain filesystems

A filesystem contains files and directories that are represented by special structures called "inodes" and "data blocks." These structures make it possible for the operating system to create and keep track of filesystems.

Data blocks — A block is a 1024-byte unit of data stored on the disk. A data block can contain either directory entries or file data. A directory entry consists of an inode number and a filename.

Inodes — An inode can be thought of as a card from a library card catalog. Each inode contains information about a file, just as a card contains information about a book, including its location, its size, the type of file, and the number of directory entries linked to the file. One important point to remember is that an inode does not contain the name of a file; directories contain the actual names. An inode contains the locations of all the data that make up a file so the operating system can collect it all when needed.

Blocks are not just stored on the hard disk. To minimize seeking data on the hard disk, recently used data blocks are held in a cache of special memory structures called buffers. These structures make the operating system more efficient. When enough data is accumulated to write out one or more full disk blocks, the buffer is "flushed" by writing its information to the disk. A minor amount of information is always lost when an outage occurs because recently changed data has not been written to the disk.

With a hard disk filled with data, inodes, directories, files, and blocks cached in memory, how does the operating system keep track of them? The answer is that all these structures maintain sufficient connectivity between files and directories to allow severed connections to be reconstructed.

One special data block, the "super" block, contains overall information about the filesystem, rather than where a particular piece of a file is located. The super block contains the information necessary to mount a filesystem and access its data. It contains the size of the filesystem, the number of free inodes, and information about free space available.

Information is read from the disk version of the super block when the filesystem is mounted and is maintained and modified in memory as activity takes place on the system. The information is written back to the disk at regular intervals by the **init** command which is always running. The **init** command calls the **sync**(ADM) command every 60 seconds, which forces the memory version of the super block and buffers to be written to disk. If the system crashes and the information stored on the disk is not reasonably up-to-date, the filesystem might be corrupted.

Causes of filesystem corruption

Corruption can affect all the structures mentioned in this section. This means that the data or the structures used to locate data can be damaged. This can occur for several reasons:

Hardware failure	Hardware failures are rare. The best way of dealing with it is to be sure that recommended diagnostic and maintenance procedures are followed conscientiously.
Program interrupts	It is possible that errors that cause a program to fail might result in the loss of some data. It is not easy to generalize about this because the range of possibilities is so large.
Human error	While it may be painful to admit it, probably the greatest cause of filesystem corruption falls under this heading. There are rules that should be followed when managing filesystems.

Rules for checking filesystems

When checking and repairing filesystems, keep in mind the following rules:

- To make sure users cannot access the filesystem you are planning to unmount, bring the system down to single-user (maintenance) mode before unmounting it.
- Unmount the filesystem with **umount**(ADM) before checking and repairing it.
- In the case of a floppy disk filesystem, unmount it before physically removing it.
- If you plan to make the filesystem available, remember to remount the filesystem after checking and repairing it.

Regular filesystem backups represent the best assurance of continued filesystem integrity.

Repairing filesystems with fsck

To check and repair filesystems, use **fsck**(ADM). The **fsck** command examines the various structures on the disk and attempts to reconcile them. Where possible, **fsck** reestablishes connections and resolves references; it "cleans" the filesystem.

Before repairing a filesystem with **fsck**, shut down the system and bring it up in single-user (maintenance) mode. To do this without completely shutting down the system, use the **su** argument to **shutdown**:

/etc/shutdown -g10 su

The first argument indicates the number of minutes before system shutdown.

NOTE The **shutdown** command calls **sync** automatically.

For more information on shutting the system down, see **shutdown**(ADM).

Once the system is in single-user mode, unmount the filesystem with **umount**(ADM). Check and repair the filesystem using the following command:

fsck ***filesystem***

Δ **sysadmsh** users select: Filesystems ⇨ Check

where ***filesystem*** is the name of the special file corresponding to the device name of the filesystem. For example, the */dev/u* device file corresponds to the */u* filesystem.

To check the root filesystem, use the **-b** argument to **fsck**:

fsck -b /dev/root

The system must be in single-user mode before running **fsck -b** on the root filesystem. This command automatically remounts the root filesystem after checking it.

NOTE The **fsck** program is actually a front-end that invokes a version of **fsck** for each filesystem type. For example, **fsck** calls a special version to repair DOS filesystems.

For example, if you bring your system back up in single-user mode after a power failure, use the **fsck** command to check the */u* filesystem before going into multiuser mode. To do this, enter the following command:

fsck /dev/u

The **fsck** program checks the filesystem and reports on its progress with the following messages:

```
** Phase 1 - Check Blocks and Sizes
** Phase 2 - Pathnames
** Phase 3 - Connectivity
** Phase 4 - Reference Counts
** Phase 5 - Check Free List
```

If a damaged file is found during any one of these phases, **fsck** asks if it should be repaired or salvaged. Enter **y** to repair a damaged file. You should always allow the system to repair damaged files even if you have copies of the files elsewhere or intend to delete the damaged files.

Note that the **fsck** command deletes any file that it considers too damaged to be repaired. You can elect for **fsck** to make the repair or not. You might choose to have **fsck** ignore an inconsistency because the problem is so severe that either you want to fix it yourself using the **fsdb**(ADM) utility, or you plan to restore your system from backups. If you cannot use **fsdb**, you must allow **fsck** to resolve the inconsistencies or the filesystem may not be usable. (**fsdb** is described in "Repairing a filesystem when fsck stops at size check" (page 571).)

You may need to run **fsck** several times before the entire filesystem is clean. For a complete list of error messages, see the **fsck**(ADM) manual page.

Summary of fsck phases

The **fsck** program scans and examines each of the structures mentioned earlier. Each phase compares components and checks that these components agree with each other.

Phase 1 checks the blocks and sizes. The **fsck** program reads the inode list to determine the 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.) When **fsck** asks whether or not to clear an inode, this means to zero out the bad information in the inode. This removes the file or directory that was associated with it. A duplicate block means that two inodes point to the same block on the disk. The **fsck** command attempts to find the original inode along with the duplicate for correction in phase 2.

Phase 2 checks the pathnames. Files removed in phase 1 must then have their directory entries removed. Phase 2 cleans up error conditions caused by improper inode status, out-of-range inode pointers, and directories that point to bad inodes as described earlier. For files with duplicate blocks found in phase 1, **fsck** wants to remove both files (this is one of the few areas where system administrator intervention is useful).

Phase 3 checks for connectivity. Phase 2 removed directories that do not point to valid files. Phase 3 reconnects files that were severed from the directory structure. Any files that are unreferenced but valid are placed in a special directory called *lost+found*. Because the directory was severed, the name of the file is lost and a number is assigned to the file in *lost+found*.

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. A *lost+found* directory is created automatically by **mkdev fs** when a new filesystem is created.

Phase 4 checks the reference counts. The **fsck** command checks the link count of each entry that survives phases 2 and 3. In some cases, files that were not pointed to under the directory structure, but still have an inode, can be relinked to the filesystem in *lost+found*.

Phase 5 checks the free list. The **fsck** command examines the free-block list maintained by the filesystem and resolves the missing or unallocated blocks allocated or removed earlier. When an inconsistency is detected, **fsck** prompts to rebuild it.

Phase 6 salvages the free list. If specified in phase 5, the system reconstructs a free block list from the altered filesystem.

Automatic filesystem check

The operating system sometimes requests a check of the filesystem when you first start it. This usually occurs after an improper shutdown (for example, after a power loss). The filesystem check repairs any files disrupted during the shutdown.

Chapter 6

Using printers

Most systems require the ability to print out data on paper. A wide variety of printing hardware or line printers are supported. Some line printers are *parallel* devices, but most are connected as *serial* devices.

To add a printer, the system administrator must:

- connect the physical hardware to the computer, then
- use the correct system commands to enable the printer for operation.

This chapter explains how to maintain printers. Consult the *Hardware Configuration Guide* for information on connecting and configuring printers.

The operating system supports serial printers which use the standard RS-232 interface and parallel printers which use the Centronics parallel interface. To find out which interface your printer uses, check your hardware documentation.

The printer spooling system

The UNIX system line printer spooling system is a collection of commands that help you to install, monitor, and efficiently control the line printers serving your system. A request to print a file is *spooled* or lined up with other printing jobs to be sent to the printer. Each print job is processed and waits its turn in line to be printed. This line of pending print jobs is called the *queue.*

When a user requests a file to be printed using the **lp**(C) command, the line printer system responds with a "request ID." This consists of the name of the printer on which the file is printed and a unique number identifying the file. With this request ID, the user can find out the status of the print request or cancel it. The **lp** options help the user to control printer output easily.

The print service performs the following functions:

- handles the task of receiving files that users want printed
- filters the files (if necessary) so they can print properly
- schedules the work of one or more printers
- starts programs that interface with the printer(s)
- keeps track of the status of jobs
- alerts you to printer problems
- keeps track of the mounting of forms
- issues error messages when problems arise

There are several terms used in this chapter to describe the operation of the print service:

device The target for **lp** output. A device is represented by a full pathname of a special device file.

printer The name assigned by the system administrator to represent a device. This name can have up to 14 characters. At different times, a printer can be associated with different devices.

class An ordered list of printers. Print requests sent to a class of printers are printed by the first available member of that class.

destination A place where print requests are sent. A destination can be a class or a printer.

Consult your computer and line printer hardware manuals for information on making the connection between your system and printing devices.

Printer configuration options

Although the default values for printer configuration are usually sufficient for most needs, there are a number of options to configure individual aspects of printer operations. These include such options as fault alerting and recovery. The following is a list of additional information that can be given to define the configuration of each printer:

- interface program
- printer type
- content types
- connection method
- character sets or print wheels

- fault alerting
- fault recovery
- use restrictions
- default printing attributes

You need to give very little of this information to add a new printer to the print service; however, the more information you provide, the better the printer is managed for you and the more efficiently the print service will run.

The descriptions in the following sections help you understand what this printer configuration information means and how it is used so that you can decide how to configure your printers. In each section, you are also shown how to specify this information when adding a printer. While you can follow each of the sections in order and correctly configure a printer in several steps, you may want to wait until you have read all of the sections before adding a printer so that you can do it in one step.

Interface program

Interface programs (also known as interface scripts) are used to initialize printers and take advantage of their particular capabilities. The *standard* interface program defines minimal capabilites common to most printers. These programs are human-readable files located in */usr/spool/lp/model*. The interface programs already present on your system are listed in Table 6-1. You can select an interface program when you install or change a printer configuration using the **sysadmsh** selections

Printers ➪ Configure ➪ Add

or

Printers ➪ Configure ➪ Modify

Table 6-1 Printer interface programs

Name	Description
1640	DASI 1640 terminal
5310	AT&T 5310/20 Matrix Printer
HPDeskJet500	Hewlett Packard DeskJet 500
HPDeskJetplus	Hewlett Packard DeskJet Plus
HPLaserJet	Hewlett Packard LaserJet
TandyDMP	Tandy DMP Printers
crnlmap	serial or parallel printer requiring newline mapped to control-linefeed

(Continued on next page)

Table 6-1 Printer interface programs
(Continued)

Name	Description
dqp10	DQP-10 Matrix Printer
dumb	dumb line printer
emulator	Tandy Printers in IBM Emulation Mode
epson	Epson serial or parallel printer
f450	DASI 450 terminal
hp	hp2631a line printer
lqp40	LQP-40 Letter Quality Printer
network	remote printing over UUCP or ethernet
network.ps	remote printing over UUCP or ethernet for PostScript printers
ph.daps	Autologic APS-5 phototypesetter
postscript	PostScript printer
pprx	Printronix line printer with parallel interface
proprinter	IBM Proprinter XL
prx	Printronix line printer
qume1155	Qume Sprint 1155 line printer
standard	standard printer interface program
ti800	Texas Instruments 855 printer

Many of these interface programs have special options available by using the **-o** option to the **lp** command. (Read the appropriate interface file for this information.) For example, the PostScript interface program includes the options listed in Table 6-2.

Table 6-2 PostScript options

Option	Description
port	prints text in portrait mode
land	prints text in landscape mode
land2	prints text in 2 page landscape mode
raw	prints a PostScript file*

* Use the raw option for PostScript files only; the postscript interface script converts text files to PostScript automatically.

You can also create your own interface scripts or customize existing ones to suit your needs. Refer to "How to write an interface program" (page 163) for more information.

Printer type

The printer type is the generic name for the printer. The print service uses the printer type to extract information about the printer from the *terminfo* database. This information describes the capabilities of the printer so that you can be warned if some of the configuration information you provide is not appropriate for the printer. The information also describes the control data to use to initialize the printer before printing a file.

NOTE Use of the *terminfo* database is not recommended. Printer capabilities are most commonly defined through interface scripts.

The printer type is the generic name for the printer. Typically, the printer type is derived from the manufacturer's name, such as "495" for the AT&T 495 Laser Printer. Examine */usr/lib/terminfo* for an appropriate entry for your "type" of printer. There might be one entry specific to your printer, or one that is generic to a type. Specify the printer type as follows:

/usr/lib/lpadmin -p ***printername*** **-T** ***printer-type***

Δ **sysadmsh** users select: Printers ⇨ Configure ⇨ Parameters

If you do not define the printer type, the default **unknown** is used. This produces empty results when the print service looks up information about the printer, so the print service cannot verify certain requests or initialize the printer.

Content types

While the printer type information tells the print service what type of printer is being added, the content type information tells the print service what types of file can be printed. Most printers can print only one type of file; for them, the content type is likely to be identical to the printer type. Some printers, however, can accept several different types of file and print their contents properly. When adding this kind of printer, you should list the names of the content types it accepts.

When a file is submitted to the print service for printing, the print service searches for a printer capable of handling the job. The print service can identify an appropriate printer through either the content-type name or the printer-type name. Therefore, you can specify either name (or no name) when submitting a file for printing.

Content-type names may look a lot like printer-type names, but you are free to choose names that mean something to you and the people using the printer. (The names *simple*, *terminfo*, or *any* are recognized as having particular meanings by the print service; be sure to use them consistently.) The names must contain no more than 14 characters and may include only letters, digits,

and underscores. If the same content type is printable by several different types of printer, you should use the same content type names when you add those printers. This makes it easier for the people using the printers because they can use the same name to identify the type of file they want printed regardless of the printing destination.

For example, several manufacturers produce printers that accept PostScript files. While these printers may need different printer types so that each can be properly initialized (assuming the initialization control sequences are different), they may all be capable of handling the same type of input file, which you call, perhaps, *PostScript*. As another example, several manufacturers produce printers that accept ANSI X3.64 defined escape sequences. However, the printers may not support all the ANSI capabilities or may support different sets of capabilities. You may want to give different content-type names for these printers to differentiate them.

You do not have to list the content types for a printer. If you do not, the printer type is used as the name of the content type the printer can handle. If you have not specified a printer type, the print service assumes the printer can print only files of content type *simple*. This may be sufficient if users pick the proper printer and make sure the files are properly prepared for the printer before they are submitted for printing.

The most common type of file on the UNIX system is known as *simple*. This file is assumed to contain just printable ASCII characters and the following control characters:

backspace	moves the printing mechanism back one space, except at the beginning of a line
tab	moves the printing mechanism to the next tab stop, which is normally every 8 columns on most printers
linefeed	moves the printing mechanism to the beginning of the next line (may require special port settings for some printers — refer to "Adjusting the printer port characteristics" (page 159) for more information)
form feed	moves the printing mechanism to the beginning of the next page
carriage return	moves the printing mechanism to the beginning of the same line (may fail on some printers)

The word "carriage" may be archaic for modern laser printers, but similar actions apply. If a printer can handle a *simple* type of file, you should include it in the content type list when you add the printer and specify the content type(s) that the printer can handle. If you do not want a printer to accept files of type *simple*, you must give an alternate list of content types that the printer

can accept. (The printer type is a good name to use if no other type is appropriate.)

Another content type name is *terminfo*. This does not refer to a particular type of file but instead refers to all the types represented in the *terminfo* database. It is not likely that any printer is capable of handling all the types listed in the database. However, this name is reserved for describing possible filter capabilities. Likewise, the content type *any* is reserved for describing the types of files a filter can accept or produce. These names should not be used as content types when adding a printer.

Specify the list of content types as follows:

/usr/lib/lpadmin -p *printername* -I *content-type-list*

Δ **sysadmsh** users select: Printers ➪ Configure ➪ Content

The ***content-type-list*** is a list of names separated by a comma or space. If you use spaces to separate the names, enclose the entire list (but not the **-I**) in quotes. If you do not define the types of files a printer can accept, the print service assumes it can take type *simple* and a type with the same name as the printer type (if the printer type is defined).

Connection method

The print service allows you to connect your printers in a variety of ways. The simplest way is to connect your printer directly to the computer. However, you may want to connect printers via a network or through a dialed modem, where they can be shared with other computers or workstations. Once you have connected the printer to the computer or connected it to a network, and connected the network to the computer, you should describe the connection method for the print service.

The default method by which printers are connected to the computer is the direct connection method. If you have used this method to connect your printer to your computer, you generally need to do only one other thing: name the connecting port.

There are two methods of making non-direct connections: through a dial-up modem or over any other type of network. The difference between a UUCP network printer and a dialup printer is that a dialup printer does not go through the UUCP spooling process; the connection is made directly and the print job prints directly on the dialed-up printer. Refer to "Configuring a network printer" (page 170) or "Configuring a dialup printer" (page 171) for more information.

Character sets or print wheels

Printers differ in the way they can print in different font styles. Some have changeable print wheels, some have font cartridges, others have preprogrammed, selectable character sets. The print service, with your help, can minimize the impact of these differences on the users of the print service.

When adding a printer, you can specify what print wheels, font cartridges, or character sets are available with the printer. Only one of these is assumed to apply to each printer. From the point of view of the print service, however, print wheels and font cartridges are the same because they require you to physically intervene and mount a new print wheel or font cartridge. Thus, for ease of discussion, only print wheels and character sets are mentioned.

When you list the print wheels or character sets available, you are assigning names to them. These names are for your convenience and the convenience of the users. Because different printers may have similar print wheels or character sets, you should use common names for all printers. This allows a person to submit a file for printing and to ask for a particular font style, without regard for which printer is used or whether a print wheel or selectable character set is used.

If the printer has mountable print wheels, you need only list their names. If the printer has selectable character sets, you need to list their names and map each one into a name or number that uniquely identifies it in the *terminfo* database. You can use the following command to determine the names of the character sets listed in the *terminfo* database:

TERM=*printer-type* tput csnm 0

printer-type is the name of the printer type in question. The name of the 0th character set (the character set obtained by default after the printer is initialized) should be printed. Repeat the command, using 1, 2, 3, and so on in place of the 0, to see the names of the other character sets. In general, the *terminfo* names should closely match the names used in the user documentation for the printer. However, because not all manufacturers use the same names, the *terminfo* names may differ from one printer type to the next.

NOTE For the print service to find the names in the *terminfo* database, you must specify a printer type. See the earlier section "Printer type."

To specify a list of print wheel names when adding a printer, enter the following command:

/usr/lib/lpadmin -p *printername* -S *print-wheel-list*

Δ **sysadmsh** users select: Printers ➪ Configure ➪ Parameters

print-wheel-list is a list of names separated by a comma or space. If you use spaces to separate the names, enclose the entire list (but not the **-S**) in quotes.

To specify a list of character set names and to map them into *terminfo* names or numbers, enter the following command:

/usr/lib/lpadmin -p *printername* -S *character-set-list*

Δ **sysadmsh** users select: Printers ➪ Configure ➪ Parameters

character-set-list is also a list of names separated by a comma or space; however, each item in the list looks like one of the following:

cs*N*=*character-setname*
character-setname1*=*character-setname2

N in the first case is a number from 0 to 63 that identifies the number of the character set in the *terminfo* database. ***character-setname1*** in the second case identifies the character set by its *terminfo* name. In either case, the name to the right of the equal sign (=) is the name you choose as an alias of the character set.

NOTE You do not have to provide a list of aliases for the character sets if the *terminfo* names are adequate. You can refer to a character set by *terminfo* name, by number, or by your alias.

For example, suppose your printer has two selectable character sets (sets #1 and #2) in addition to the standard character set (set #0). The printer type is *5310*. You enter the following commands to determine the names of the selectable character sets:

```
TERM=5310 tput csnm 1
english
TERM=5310 tput csnm 2
finnish
```

The words "english" and "finnish" are the output of the commands, the names of the selectable character sets. You feel that the name "finnish" is adequate for referring to character set #2, but better names are needed for the standard set and set #1. You enter the following command to define synonyms:

/usr/lib/lpadmin -p *printername* -S "cs0=american, english=british"

Δ **sysadmsh** users select: Printers ➪ Configure ➪ Parameters

If you do not list the print wheels or character sets that can be used with a printer, then the print service assumes the following: a printer that takes print wheels has only a single, fixed print wheel, and people cannot ask for a special print wheel when using the printer. Also, a printer that has selectable character sets can take any **cs***N* name or *terminfo* name known for the printer.

Alerting to mount a print wheel

If you have printers that take changeable print wheels and you have listed the print wheels allowed on each, then users can submit a print request to use a particular print wheel. However, until it is mounted a request for a print wheel stays queued and is not printed. (Refer to "Mounting a form or print wheel" (page 145) for more information.) You could periodically monitor the number of print requests pending for a particular print wheel, but the print service provides an easier way. You can ask to be alerted when the number of requests waiting for a print wheel has exceeded some threshold.

You can choose one of several ways to receive an alert:

- You can receive an alert via electronic mail. See **mail**(C) for a description of the **mail** command.
- You can receive an alert written to whatever terminal you are logged in on. You can enable the writing of messages to your terminal with the **mesg**(C) command. (The most convenient way to do this is to add the command **mesg y** in your *.profile*, *.cshrc*, *.login*, or *.kshrc* file.)
- You can receive an alert through a program of your choice.
- You can receive no alerts.

NOTE If you elect to receive no alerts, you are responsible for checking whether the proper print wheel is mounted.

In addition to the method of alerting, you can also set the number of requests that must be queued before you are alerted, and you can arrange for repeated alerts every few minutes until the print wheel is mounted. You can choose the rate of repeated alerts, or you can choose to receive only one alert per print wheel.

To arrange for alerting to the need to mount a print wheel, enter one of the following commands:

/usr/lib/lpadmin -S ***print-wheelname*** **-A mail -Q** ***integer*** **-W** ***minutes***
/usr/lib/lpadmin -S ***print-wheelname*** **-A write -Q** ***integer*** **-W** ***minutes***
/usr/lib/lpadmin -S ***print-wheelname*** **-A** ***'command'*** **-Q** ***integer*** **-W** ***minutes***
/usr/lib/lpadmin -S ***print-wheelname*** **-A none**

Δ **sysadmsh** users select: Printers ➪ Auxiliary ➪ Alert

The first two commands direct the print service to send you a mail message or write the message directly to your terminal, respectively, for each alert. The third command directs the print service to run ***command*** for each alert. The shell environment currently in effect when you enter the third command is saved and restored for the execution of ***command***; this includes the environment variables, user and group IDs, and current directory. The fourth command directs the print service to never send you an alert when the print wheel needs to be mounted. ***integer*** is the number of requests that need to be waiting for the print wheel, and ***minutes*** is the number of minutes between repeated alerts.

NOTE If you want mail sent or a message written to another person, you will have to use the third command listed. Use the **-A 'mail *user-name*'** or **-A 'write *user-name*'** option.

Once you start receiving repeated alerts, you can direct the print service to stop sending you alerts for the current case by giving the following command:

/usr/lib/lpadmin -S *print-wheelname* -A quiet

Δ **sysadmsh** users select: Printers ⇨ Auxiliary ⇨ Alert

Once the print wheel is mounted and unmounted again, alerts start again if too many requests are waiting. Alerts also start again if the number of requests waiting falls below the **-Q** threshold and then rises up to the **-Q** threshold again, as when waiting requests are canceled or if the type of alerting is changed.

If ***print-wheelname*** is *all* in any of the commands above, the alerting condition applies to all print wheels for which an alert has already been defined.

If you do not define an alert method for a print wheel, you do not receive an alert for it. If you do define a method but do not give the **-W** option, you are alerted once for each occasion.

Fault alerting

The print service provides a framework for detecting printer faults and alerting you. Faults can range from simple problems, such as running out of paper or ribbon or needing to replace the toner, to more serious faults, such as a local power failure or printer failure. The range of fault indicators is also broad, ranging from dropping carrier (the signal that indicates that the printer is online) to sending an XOFF, or a message. Only two classes of printer fault indicators are recognized by the print service itself: a drop in carrier and an XOFF not followed in reasonable time by an XON. However, you can add filters that can recognize any other printer fault indicators and rely on the print service to alert you to a fault when the filter detects it.

NOTE Refer to "Using forms and filters" (page 155) to add a filter. For a description of how a filter should let the print service know a fault has occurred, see the **lpforms**(ADM) manual page.

You can choose one of several ways to receive an alert to a printer fault:

- You can receive an alert via electronic mail. See **mail**(C) for a description of the **mail** command.
- You can receive an alert written to the terminal on which you are logged in (any terminal). See **write**(C) for a description of the **write** command.
- You can receive an alert through a program of your choice.
- You can receive no alerts.

NOTE If you elect to receive no alerts, you need a way of finding out about the faults and fixing them; the print service does not continue to use a printer that has a fault.

In addition to the method of alerting, you can also arrange for repeated alerts every few minutes until the fault is cleared. You can choose the rate of repeated alerts, or you can choose to receive only one alert per fault.

NOTE Without a filter that provides better fault detection, the print service cannot automatically determine when a fault has been cleared except by trying to print another file. It assumes that a fault is cleared when it successfully prints a file. Until that time, if you have asked for only one alert per fault, you do not receive another alert. If after you have fixed a fault, but before the print service has tried printing another file, the printer faults again, or if your attempt to fix the fault did not succeed, you are not notified. Receiving repeated alerts per fault or requiring manual reenabling of the printer overcomes the problem (refer to "Fault recovery" (page 133) for more information).

To arrange for alerting to a printer fault, enter one of the following commands:

/usr/lib/lpadmin -p ***printername*** **-A mail -W** ***minutes***
/usr/lib/lpadmin -p ***printername*** **-A write -W** ***minutes***
/usr/lib/lpadmin -p ***printername*** **-A** ***'command'*** **-W** ***minutes***
/usr/lib/lpadmin -p ***printername*** **-A none**

Δ **sysadmsh** users select: Printers ➪ Configure ➪ Errors

The first two commands direct the print service to send you a mail message or write the message directly to your terminal, respectively, for each alert. The third command directs the print service to run ***command*** for each alert. The shell environment currently in effect when you enter the third command is saved and restored for the execution of ***command***. The environment includes environment variables, user and group IDs, and current directory. The

minutes is the number of minutes between repeated alerts. The fourth command directs the print service not to send you an alert when a fault occurs.

NOTE If you want mail sent or a message written to another person when a printer fault occurs, use the third command. Use the option:

-A 'mail *username*'

or

-A 'write *username*'

Once a fault occurs and you start receiving repeated alerts, you can direct the print service to stop sending you alerts for the current fault by giving the following command:

/usr/lib/lpadmin -p *printername* -A quiet

Δ **sysadmsh** users select: Printers ⇨ Configure ⇨ Errors

If ***printername*** is *all* in any of the commands above, the alerting condition applies to all printers.

If you do not define an alert method, you receive mail once for each printer fault. If you do define a method but do not give the **-W** option, you are alerted once for each fault.

Fault recovery

Once a printer fault is detected and you are alerted, you will probably fix the fault and get the printer ready for printing. When the printer is ready for printing again, the print service recovers in one of three ways:

- continues printing at the top of the page where printing stopped,
- restarts printing at the beginning of the print request that was active when the fault occurred, or
- waits for you to tell the print service to re-enable the printer.

NOTE The ability to continue printing at the top of the page where printing stopped requires the use of a filter that can wait for a printer fault to be cleared before resuming properly. Such a filter probably has to have detailed knowledge of the control sequences used by the printer so it can keep track of page boundaries and know where in a file printing stopped. The default filter used by the print service cannot do this. If a proper filter is not being used, you are notified in an alert if recovery cannot proceed as you want.

To specify the way the print service should recover after a fault has been cleared, enter one of the following commands:

/usr/lib/lpadmin -p ***printername*** **-F continue**
/usr/lib/lpadmin -p ***printername*** **-F beginning**
/usr/lib/lpadmin -p ***printername*** **-F wait**

Δ **sysadmsh** users select: Printers ⇨ Configure ⇨ Errors

These direct the print service, respectively, to continue at the top of the page, restart from the beginning, or wait for you to enter an **enable** command to re-enable the printer (refer to "Enabling and disabling printers" (page 143) for information on the **enable** command).

If you do not specify how the print service is to resume after a printer fault, it tries to continue at the top of the page where printing stopped, or failing that, at the beginning of the print request.

If the recovery is **continue** but the interface program does not stay running so that it can detect when the printer fault was cleared, printing is attempted every few minutes until it succeeds. You can force the print service to retry immediately by issuing an **enable** command.

Restricting user access to a printer

You can limit the use of a printer to a subset of all people on your computer. You may want to do this, for instance, if a printer is being set aside for printing sensitive information and only a subset of the people are allowed to print sensitive information, or if use of a high-quality printer incurs expenses not all people are authorized to incur.

The print service uses the list of users allowed or denied for a printer to restrict use of the printer. The print service refuses a user's request to print a file on a printer he or she is not allowed to use.

The method of listing the users allowed or denied for a printer is similar to the method used to list users allowed or denied access to the **cron** and **at** facilities. Briefly, the rules are as follows:

- An allow list contains those users allowed to use the printer. A deny list contains those users denied access to the printer.
- If the allow list is not empty, the deny list is ignored. If the allow list is empty, the deny list is used. If both lists are empty, there are no restrictions on who can use the printer.
- Putting "any" or "all" into the allow list allows everybody to use the printer; putting "any" or "all" into the deny list denies everybody, except the user *lp* and the super user (*root*).

You can add names of users to either list using one of the following commands:

/usr/lib/lpadmin -p ***printername*** **-u allow:***user-list*
/usr/lib/lpadmin -p ***printername*** **-u deny:***user-list*

Δ **sysadmsh** users select: Printers ⇨ Configure ⇨ Users

user-list is a list of names of users separated by a comma or space. If you use spaces to separate the names, enclose the entire list (including **allow:** or **deny:** but not the **-u**) in quotes. The first command adds the names to the allow list and removes them from the deny list. The second command adds the names to the deny list and removes them from the allow list. Using **allow:all** allows everybody; using **deny:all** denies everybody.

If you do not add user names to the allow or deny lists, the print service assumes that everybody can use the printer.

Default printing attributes

When a user submits a request to print a file, the page size, character pitch, and line pitch (that is, print spacing) are normally determined from the form that it is printed on. If the user does not require a form, he or she can give the page size and print spacing to use. However, if the user gives neither a form to use nor the page size and print spacing, defaults are used.

You can set the defaults for each printer. This can also serve to make submitting a print request easier, by designating different printers as having different default page sizes or print spacing. Users then simply route their file to the appropriate printer to get the style output they want. For example, you can have one printer dedicated to printing wide (132 column) output, another printing normal (80 column by 66 lines) output, yet another printing letter quality (12 characters per inch, 8 lines per inch).

You can independently specify four default settings: page width, page length, character pitch, and line pitch. You can scale these to fit your needs. The first two can be given in columns and lines, inches, or centimeters. The last two can be given as characters and lines per inch or per centimeter. In addition, the character pitch can be specified as **pica** for 10 characters per inch (cpi), **elite** for 12 cpi, or **compressed** for the maximum cpi the printer can provide (up to a limit of 30 cpi).

Set the defaults using one or more of the following commands:

/usr/lib/lpadmin -p ***printername*** **-o width=***scaled-number*
/usr/lib/lpadmin -p ***printername*** **-o length=***scaled-number*
/usr/lib/lpadmin -p ***printername*** **-o cpi=***scaled-number*
/usr/lib/lpadmin -p ***printername*** **-o lpi=***scaled-number*

Δ **sysadmsh** users select: Printers ⇨ Configure ⇨ Parameters

Add the letter "i" to ***scaled-number*** to indicate inches, or the letter "c" to indicate centimeters. The letter "i" for character pitch (cpi) or line pitch (lpi) is redundant. You can also give **pica**, **elite**, or **compressed** instead of a number for the character pitch.

If you do not provide defaults, the page size and print spacing are those available when the printer is initialized. You can find out what the defaults are by first defining the printer configuration without providing your own defaults, then using the **lpstat** program to display the printer configuration. The command

lpstat -p ***printername*** **-l**

reports the default page size and print spacing. If you have not provided the defaults, the reported defaults are calculated from the *terminfo* database entry for the printer. Obviously, this requires you to have provided a printer type in the printer configuration.

Print service command summary

The print service has four user-level commands, which are shown in Table 6-3.

Table 6-3 User print service commands

Command	Description
cancel(C)	cancels a request for a file to be printed
lp(C)	sends a file or files to a printer
lpstat(C)	reports the status of the print service
lprint(C)	prints from printer attached to a terminal

In addition to sending requests to the print service system, checking the status of requests, and canceling requests, users can be given the ability to disable and enable a printer. The idea is that if a user finds a printer is malfunctioning in some way, it should not be necessary to call the administrator to turn the printer off. On the other hand, it may not be reasonable in your printing environment to allow regular users to disable a printer. You can control

whether other users have access to the two commands shown in Table 6-4 by assigning or revoking the **printerstat** authorization. Refer to "Changing user authorizations" (page 74) for more information.

Table 6-4 Privileged print service commands

Command	Description
disable(C)	deactivates the named printer(s)
enable(C)	activates the named printer(s)

A separate set of commands available for the administrator is shown in Table 6-5. These commands are found in the */usr/lib* directory. If you expect to use them frequently, you might find it convenient to include that directory in your **PATH** variable. To use the administrative commands, you must be logged in as either *root* or have the **lp** authorization. Refer to "Changing user authorizations" (page 74) for more information. (**lpsched** must be run by *root*.)

Note that all these commands are accessible under the **sysadmsh** Printers selection. You will also probably need to use the commands for disabling and enabling a printer and the rest of the commands described earlier in this section.

Table 6-5 Administrative print service commands

Command	Description
/usr/lib/accept	Permits job requests to be queued for a specified destination.
/usr/lib/reject	Prevents jobs from being queued for a specified destination. Described on the **accept**(ADM) manual page.
/usr/lib/lpadmin	Sets up or changes printer configurations.
/usr/lib/lpfilter	Sets up or changes filter definitions.
/usr/lib/lpforms	Sets up or changes preprinted forms (Enter **/usr/lib/lpadmin** to mount a form).
/usr/lib/lpmove	Moves output requests from one destination to another. Described on the same manual page as **lpsched**(ADM).

(Continued on next page)

Table 6-5 Administrative print service commands
(Continued)

Command	Description
/usr/lib/lpsched	Starts the print service.
/usr/lib/lpshut	Stops the print service. Described on the **lpsched**(ADM) manual page.
/usr/lib/lpusers	Sets or changes the default priority and priority limits the users of the print service can request.

These commands are also accessed through the **sysadmsh** Printers selection, which is much easier than the complex syntax of the print service commands.

Adding a local printer

The operating system also supports the use of local printers attached to the AUX or PRINT port on the back of many serial terminals. These printers are connected via standard RS-232 connections and can significantly reduce the load on shared system resources. The **lprint**(C) command is used to print files on a local printer, but the terminal must be properly configured for the command to work. To add a printer connected to the AUX or PRINT port on the back of a terminal and use it for local printing, follow this procedure:

1. Connect your local serial printer to the AUX port on your terminal with a standard RS-232 cable with pins 2, 3, and 7 (minimum) connected. Make sure the printer is powered on and is online. (If the terminal supports pass-through mode to the parallel port, a parallel port may be used.)
2. Log in to the UNIX 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. For the **lprint** command to work, **lprint** needs to know how to start and stop local printing for each specific terminal. **lprint** looks in the file */etc/termcap* to find two terminal attributes: PN (start printing) and PS (stop printing). These are escape sequences that must be sent to the terminal to control local printing. Very few terminals have these attributes defined in their *termcap* entries. Use a text editor (such as **vi**(C)) to examine the */etc/termcap* file. (*/etc/termcap* can also be an alternate file, as defined by the **TERMCAP** variable.) Search for the entry for your terminal. For example, if your terminal is a Wyse 60, you would search for "wyse60".

The *termcap* entry for the Wyse 60 appears in Example 6-1.

Example 6-1 Wyse 60 termcap entry

```
w7|wy60|wyse60|Wyse WY-60 with 80 column/24 line screen in wy60 mode:\
    :is=\E'\072\Ee(\EO\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. With other terminals, you must add a line containing these two attributes to the */etc/termcap* entry for your terminal. The line you add has the form:

:PN=*start sequence*:PS=*stop sequence:*

5. 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 an example of the sequence to enable auxiliary printing, the code to switch the port on for a Wyse 60 terminal is:

   ⟨Esc⟩ d #

   And the code to turn it off again is:

   ⟨Ctrl⟩t

6. These keystrokes must be translated into *termcap* format before inserting them into the *termcap* file. *termcap* uses the codes in Table 6-6 to represent keystrokes.

Table 6-6 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

To use a control sequence, use the caret (^) symbol, not the ⟨Ctrl⟩ key. For example, ⟨Ctrl⟩**x** would be represented by ^x. In addition, characters can be represented by their octal codes (see **ascii**(M)), and the caret and backslash characters represented by \^ and \\, respectively. Entries for *termcap* attributes must be separated by a colon (:). (See **termcap**(M) for more details.)

Recall that the *termcap* attributes for starting and ending printing are PN and PS. Using the table above the *termcap* entry for the Wyse 60 keystrokes ⟨Esc⟩ **d #** (start printing, PN) and ⟨Ctrl⟩**t** (stop printing, PS) looks like this:

```
:PN=\Ed#:PS=^T:\
```

7. For a terminal missing these entries, you simply insert a modified version of the above line into the *termcap* entry for the terminal. (You must be certain to insert the line within the entry for your terminal; do not add it as the first line or the last line.) 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.)

   NOTE You must be logged in as *root* to edit */etc/termcap*. We recommend that you copy the original file to another name in case you make an error. You can also extract the file again from your distribution using **custom**(ADM).

8. Once you have added the PN and PS entries, log out and back in again to activate the new *termcap* entry.
9. Use the following command to print the file ***filename*** on your 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.
10. If your file is printed on the screen instead of 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.)

    NOTE You can eliminate carriage return delays by setting the environment variable **CRDELAY** to "N". This should be done only if you are running a fast printer.

11. If, when printing with **lprint**, everything prints on one line, you should set the environment variable **FORMS** to "X". In the Bourne or Korn shells (**sh** or **ksh**), use:

    FORMS=X; export FORMS

    In the C shell (**csh**), use:

    setenv FORMS X

    These commands may be placed in the */etc/profile* and */etc/cshrc* files respectively, in which case they will affect all users. They also may be placed in individual users' *.profile* and/or *.cshrc* files.

    This environment setting turns on **lprint**'s "transparent" mode. In this mode **lprint** does not perform special processing of carriage returns, line feeds, form feeds, or tabs. This means that your terminal's **stty** settings must match those of the printer.

12. If the terminal and printer require different **stty** settings, you need to create a shell script (this step is unnecessary in most cases). Create a new 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
```

    Change the permissions on the file as follows:

    chmod 755 lprints

    Place this in an appropriate directory (for example: */usr/bin*) and execute it in place of **lprint**.

    Each user can now store the **stty** settings required for their local printer in the environment variable **LPRINTSTTY**, or system-wide values may be set in */etc/profile* and */etc/cshrc*, for example:

```
LPRINTSTTY="ixon ixoff -ixany onlcr"; export LPRINTSTTY
```

Starting and stopping the print service

Under normal operation, you should never have to start or stop the print service manually. It automatically starts each time the system goes into multi-user mode. However, if you need to stop the print service without stopping the operating system as well, you can do so by following the procedure described in the next section.

Stopping the print service causes all printing to cease within seconds. Any print requests that have not finished printing are printed in their entirety after the print service restarts. The printer configurations, forms, and filters in effect when the print service is stopped are restored after it is restarted.

NOTE To start and stop the print service manually, you must be logged in as *root*.

Manually stopping the print service

To stop the print service manually, enter the following command:

/usr/lib/lpshut

Δ **sysadmsh** users select: Printers ⇨ Schedule ⇨ Stop

This message is displayed:

```
Print services stopped
```

All printing ceases within a few seconds. If you try to stop the print service when it is not running, you see the message:

```
Print services already stopped
```

NOTE Jobs can appear to pass through a printer that is not online. If a printer is not online or operating properly, you should disable the printer.

Manually starting the print service

To restart the print service manually, enter the following command:

/usr/lib/lpsched

Δ **sysadmsh** users select: Printers ⇨ Schedule ⇨ Begin

This message is displayed:

```
Print services started
```

It may take a minute or two for the printer configurations, forms, and filters to be reestablished before any saved print requests start printing. If you try to restart the print service when it is already running, you see the message:

```
Print services already active
```

NOTE You do not have to stop the print service to change printer configurations or to add forms or filters.

Canceling a print request

To cancel a printout you have requested, use the **cancel**(C) command. When you request a printout, the system displays a request ID for your job. For example, if you send a job to a printer named "laser," the UNIX system displays the request ID as:

```
request id is laser-number
```

where ***number*** is the number assigned to your job. To cancel the job before it begins printing, use the following command:

cancel laser-*number*

Δ **sysadmsh** users select: Printers ⇨ Request ⇨ Cancel

The printout is canceled.

Most systems print quickly, so a **cancel** command must be used promptly to have any effect.

Enabling and disabling printers

The **enable** command allows **lpsched** to print files on printers. A printer can accept requests for printing after the **accept** command is given for it, but to print the files, the **enable** command must be given as well.

For example, to enable a printer named "daisy," enter:

enable daisy

Δ **sysadmsh** users select: Printers ⇨ Schedule ⇨ Enable

You can disable printers with the **disable** command. The scheduler, **lpsched**, does not send printing requests to disabled printers regardless of their status with respect to the **accept** command. The **disable -r** option allows you to include a message about why the printer was disabled in the output for **lpstat -t**.

For example, to disable a printer named "laser" because of a paper jam, enter:

disable -r "paper jam" laser

Users requesting the status of "laser" with the command **lpstat -plaser** receive the following message:

```
printer laser disabled since Dec 5 10:15
paper jam
```

For more information on these two commands, see the **enable**(C) and **disable**(C) manual pages.

Adding a printer to a class

It is occasionally convenient to treat a collection of printers as a single class. The benefit is that a person can submit a file for printing by a member of a class, and the print service picks the first printer in the class that it finds free. This allows faster turnaround, as printers are kept as busy as possible.

Classes are not needed if the only purpose is to allow a user to submit a print request by type of printer. The **lp -T** ***type*** command lets a user submit a file and specify its type. The first available printer that can handle the type of file prints the file.

The print service avoids using a filter, if possible, by choosing a printer that can print the file directly over one that needs it filtered first.

One use of classes is to put into a class a series of printers that should be used in a particular order. If you have a high-speed printer and a low-speed printer, for instance, you probably want the high-speed printer to handle as many print requests as possible, with the low-speed printer reserved for use when the other is busy. Because the print service always checks for an available printer in the order that the printers were added to a class, you could add the high-speed printer to the class before the low-speed printer and let the print service route print requests in the order you wanted.

Add a printer to a class using the following command:

/usr/lib/lpadmin -p ***printername*** **-c** ***classname***

Δ **sysadmsh** users select: Printers ⇨ Configure ⇨ Modify

If the class ***classname*** does not exist yet, it is created. (Use **lpstat -c** to get a list of all classes and their printers.)

NOTE Class names and printer names must be unique. This allows a user to specify the destination for a print request without having to know whether it is a class of printers or a single printer. Thus, you can not have a class and printer with the same name.

A printer does not belong to any class until you add it to one.

Setting the system default destination

You can define the printer or class used to print a file when the user has not explicitly asked for a particular destination and has not set the **LPDEST** shell variable. The printer or class must already exist first.

Make a printer or class the default destination by entering the following command:

/usr/lib/lpadmin -d ***printername*** or ***classname***

Δ **sysadmsh** users select: Printers ➪ Configure ➪ Default

If you later decide that there should be no default destination, enter a null ***printername*** or ***classname*** as in the following command:

/usr/lib/lpadmin -d

Δ **sysadmsh** users select: Printers ➪ Configure ➪ Default

If you do not set a default destination, there is none. Users must explicitly name a printer or class in each print request, or they have to set the **LPDEST** shell variable with the name of a destination.

For C-shell:

setenv LPDEST ***printer***

For Bourne or Korn shells:

LPDEST=*printer* **; export LPDEST**

Users can also place these commands in their *.login* and *.profile* files, respectively.

Mounting a form or print wheel

NOTE Refer to "Using forms and filters" (page 155) for information about preprinted forms.

Before the print service starts to print files that need a preprinted form or print wheel, you have to mount it on a printer. (Print wheels are used on older, so-called "daisy wheel" impact printers that use small wheels with the print characters around the perimeter.) If alerting has been set on the form or print

wheel, you are alerted when enough print requests are queued for it to be mounted.

When you mount a form, you may wish to see if it is lined up properly. If an alignment pattern is registered with the form, you can ask that this be repeatedly printed until you have adjusted the printer so that the alignment pattern looks correct.

Mounting a form or print wheel involves first loading it onto the printer and then telling the print service that it is mounted. Because it is difficult to do this on a printer that is currently printing and because the print service continues to print files not needing the form on the printer, you will probably have to disable the printer first. Thus, the proper procedure is as follows:

1. Disable the printer using the **disable** command.
2. Mount the new form or print wheel as described later in this section.
3. Re-enable the printer using the **enable** command. Refer to "Enabling and disabling printers" (page 143) for more information on the **enable** and **disable** commands.

After loading the new form or print wheel into the printer, enter the following command to tell the print service to mount it. (This command is shown on two lines for readability; it must be entered as one line.)

/usr/lib/lpadmin -p ***printername*** **-M -S** ***print-wheelname*** **-f** ***formname*** **-a -o filebreak**

Δ **sysadmsh** users select: Printers ➪ Auxiliary ➪ PPforms ➪ Configure

Leave out **-S** ***print-wheelname*** if you are mounting just a form, or leave out the **-f** ***formname*** **-a -o filebreak** if you are mounting just a print wheel.

If you are mounting a form, you are asked to press the ⟨Return⟩ key before each copy of the alignment pattern is printed. After the pattern is printed, you can adjust the printer and press the return key again. If no alignment pattern is registered, you are not asked to press the key. You can drop the **-a** and **-o filebreak** options if you do not want to bother with the alignment pattern.

The **-o filebreak** option tells the print service to add a *formfeed* after each copy of the alignment pattern. The actual control sequence used for the *formfeed* depends on the printer involved and is obtained from the *terminfo* database. If the alignment pattern already includes a formfeed, leave out the **-o filebreak** option.

If you want to unmount a form or print wheel, use the following command:

/usr/lib/lpadmin -p ***printername*** **-M -S none -f none**

Δ **sysadmsh** users select: Printers ➪ Auxiliary ➪ PPforms ➪ Remove

Leave out **-S none** if you just want to unmount a form; leave out **-f none** if you just want to unmount a print wheel.

Until you mount a form on a printer, only print requests that do not require a form are sent to it. Likewise, until you mount a print wheel on a printer, only print requests that do not require a particular print wheel are sent to it.

Removing a printer or class

You can remove a printer or class if it has no pending print requests. If there are pending requests, you have to first move them to another printer or class using the **lpmove** command, or remove them using the **cancel** command.

Removing the last remaining printer of a class automatically removes the class as well. However, the removal of a class does not cause the removal of printers that were members of the class. If the printer or class removed is also the system default destination, the system no longer has a default destination.

To remove a printer or class, enter the following command:

/usr/lib/lpadmin -x ***printername*** or ***classname***

Δ **sysadmsh** users select: Printers ➪ Configure ➪ Remove

If all you want to do is remove a printer from a class but not delete the printer, enter the following command:

/usr/lib/lpadmin -p ***printername*** **-r** ***classname***

Δ **sysadmsh** users select: Printers ➪ Configure ➪ Modify

Managing the printing load

Occasionally, you may need to stop accepting print requests for a printer or move print requests from one printer to another. There are various reasons for doing this, such as the following:

- The printer needs periodic maintenance.
- The printer is broken.
- The printer was removed.
- The configuration was changed so that the printer can be used differently.
- Too many large print requests are queued for one printer and should be evenly distributed.

If you are going to make a big change in the way a printer is used, such as stopping its ability to handle a certain form, changing the print wheels available for it, or disallowing some people from using it, print requests that are

currently queued for printing on it must be moved or canceled. The print service attempts to find alternate printers, but only if the user does not care which printer is used. Such requests are not automatically moved; if you do not move them first, the print service cancels them.

If you decide that a printer is to be taken out of service, its configuration is to be changed, or it is too heavily loaded, you can move print requests from it and reject additional requests for it. Use the **lpmove** and **reject** commands for this. If you do reject requests for a printer, you can later accept requests using the **accept** command.

Rejecting requests for a printer or class

To stop accepting any new requests for a printer or a class of printers, enter the following command:

/usr/lib/reject -r ***"reason" printername*** or ***classname***

Δ **sysadmsh** users select: Printers ➪ Schedule ➪ Reject

You can reject requests for several printers or classes in one command by listing their names on the same line, separating the names with spaces. The ***reason*** is displayed whenever anyone tries to print a file on the printer. You can omit this option if you do not want to give a reason.

Although the **reject** command stops any new print requests from being accepted, it does not move or cancel any requests currently queued for the printer. These continue to print as long as the printer is enabled.

Accepting requests for a printer or class

The **accept** command allows printers or classes of printers to accept print requests made with the **lp** command. You can allow a printer to accept requests after it has been properly configured.

After the condition that led to denying requests is corrected or changed, enter the following command to start accepting new requests:

/usr/lib/accept ***printername*** or ***classname***

Δ **sysadmsh** users select: Printers ➪ Schedule ➪ Accept

Again, you can accept requests for several printers or classes in one command by listing their names on the same line. You will always have to use the **accept** command for a new printer or class after you have added it because the print service does not initially accept requests for new printers or classes.

Moving requests to another printer

If you have to move requests from one printer or class to another, enter one of the following commands:

/usr/lib/lpmove ***request-id printername***
/usr/lib/lpmove ***printername1 printername2***

Δ **sysadmsh** users select: Printers ⇨ Request ⇨ Move

You can give more than one request ID before the printer name in the first command. The first command moves the listed requests to the named printer. The latter command moves all requests currently queued for the first printer to the second printer. When the latter command is used, the print service also no longer accepts requests for the first printer (this has the same effect as the **reject** command).

lpmove, accept and reject examples

Here are some examples of how you might use **lpmove, accept**, and **reject**:

Example 1
You decide to change the ribbon on printer "printer1" and perform some preventive maintenance. You want to move all the requests for printer "printer1" to printer "printer2". After the requests are moved, the print service no longer accepts requests for "printer1" (this has the same effect as a **reject printer1** command issued after the **lpmove** command).

/usr/lib/lpmove printer1 printer2

Now you can disable the printer and start working on it. When you are finished, you can bring it back into service with the following command:

/usr/lib/accept printer1

At this point, if you had disabled the printer you should re-enable it. Refer to "Enabling and disabling printers" (page 143) for more information.

Example 2
You notice that someone has queued several large files for printing on the printer "laser1". Meanwhile "laser2" is currently idle because no one had queued requests for it. You'll move the two biggest requests, "laser1-23" and "laser1-46" to "laser2", and you reject any new requests for "laser1" for the time being.

/usr/lib/lpmove laser1-23 laser1-46 laser2
/usr/lib/reject -r "too busy--will reopen later" laser1

Example 3
You want to prevent printing requests from being routed to "printer4" because of repairs:

/usr/lib/reject -r "printer4 needs repair" printer4

A user who requests a file to be printed on "printer4" receives the following message:

```
UX:lp: ERROR: Requests for destination "printer 4" aren't
              being accepted

    TO FIX: Use the "lpstat -a" command to see
            why this destination is not accepting requests
```

To find out the acceptance status of printing destinations, enter:

lpstat -a

The output looks like this:

```
printername accepting requests since date time
printer4 not accepting requests since date time
     printer4 needs repair
```

Managing queue priorities

The print service provides a simple priority mechanism that people can use to adjust the position of a print request in the queue. Each print request can be given a priority level by the person who submits it; this is a number from 0 to 39, with *lower* numbers indicating *higher* levels of priority. Requests with higher priority (smaller numbers) are placed ahead of requests with lower priority (larger numbers).

In this way, if you decide that your print request is of too low a priority, you can set a higher priority (lower value) when you submit the file for printing. If you decide that your print request is of too high a priority, you can set a lower priority (higher value) when you submit the file for printing.

A priority scheme this simple does not work if there are no controls on how high one can set the priority. You can define the following characteristics of this scheme:

- Each user can be assigned a priority limit. One cannot submit a print request with a priority higher than his or her limit, although one can submit a request with a lower priority.
- A default priority limit can be assigned for the balance of users not assigned a personal limit.

- A default priority can be set. This is the priority given to print requests to which the user does not assign a priority.

By setting the characteristics according to your needs, you can prevent lower priority printing tasks (such as regular printing by most staff members) from interfering with higher priority printing tasks (such as payroll check printing by the accounting staff).

You may find that you want a critical print request to print ahead of any others, perhaps even if it has to preempt the currently printing request. You can have the print service give **immediate** handling to a print request and put on **hold** another print request. This lets the urgent print request print and delays the current print request until you have it **resume**d.

The **lpusers** command lets you assign both priority limits for users and priority defaults. In addition, you can use the **lp -i** ***request-id*** **-H hold** and **lp -i** ***request-id*** **-H immediate** commands to put a request on hold or move it up for immediate printing, respectively. These commands are discussed in detail in the sections that follow.

Setting priority limits

To set a user's priority limit, enter the following command:

/usr/lib/lpusers -q ***priority-level*** **-u** ***username***

You can set the limit for a group of users by listing their names after the **-u** option. Separate multiple names with a comma or space (enclose the list in quotes if you use a space). The ***priority-level*** is a number from 0 to 39. As mentioned before, the lower the number, the higher the priority, or, in this case, the priority limit.

If you want to set the priority limit for all other users, enter the following command:

/usr/lib/lpusers -d ***priority-level***

Δ **sysadmsh** users select: Printers ⇨ Priorities ⇨ Default

This sets the default limit; the default applies to those people who have not been given a personal limit using the earlier **lpusers** command.

If you later decide that someone should have a different priority limit, just re-enter the first command above with a new limit. If you decide that someone with a personal limit should have whatever the default limit is, enter the following command:

/usr/lib/lpusers -u *username*

Δ **sysadmsh** users select: Printers ➪ Priorities ➪ Remove

Again, you can do this for more than one person at a time by giving a list of names. Using the **lpusers** command with just the **-u** option puts the users in the *default limit* category.

If you do not set a default limit, people without personal limits are limited to priorities in the range of 20 to 39.

Setting a default priority

You can set the default priority that should be assigned to those print requests submitted without a priority. Use the following command:

/usr/lib/lpusers -q *priority-level*

Δ **sysadmsh** users select: Printers ➪ Priorities ➪ Highest

Do not confuse this default with the *default limit*. This default is applied when a user does not give a priority; the *default limit* is applied if you have not assigned a limit for a user — it is used to limit the user from giving too high a priority.

NOTE If the default priority is greater than the limit for a user, the limit is used instead.

If you do not set a default priority, the print service uses the default of 20.

Examining the priority limits and defaults

You can examine all the settings you have assigned for priority limits and defaults by entering the following command:

/usr/lib/lpusers -l

Δ **sysadmsh** users select: Printers ➪ Priorities ➪ List

Moving a request around in the queue

Once a user has submitted a print request, you can move it around in the queue to some degree. For example, you can:

- adjust the priority to any level regardless of the limit for the user

- put it on hold and let other requests print ahead of it
- put it at the head of the queue for immediate printing

You use the regular **lp** user command to do each of these.

Changing the priority for a request

Print requests that are still waiting to print can be reassigned a new priority. This repositions the request in the queue, putting it ahead of lower priority requests but behind any others at the same or higher priority. The priority limit assigned to the user (or the default priority limit) has no effect because you override this limit as the administrator.

Enter the following command to change the priority of a request:

lp -i ***request-id*** **-q** ***new-priority-level***

You can change only one request at a time with this command. If a request is already printing, you cannot change its priority.

Putting a request on hold

Any request that has not finished printing can be put on hold. You can stop its printing, if it currently is printing, and keep it from printing until you resume it. Another user, however, cannot resume a print request that you put on hold.

Enter the following command to place a request on hold:

lp -i ***request-id*** **-H hold**

Enter the following command to resume the request:

lp -i ***request-id*** **-H resume**

Once resumed, a request continues to move up the queue and will print. If the request was printing when you held it, it is restarted and becomes the next request to print. Normally the request starts printing from the beginning, with page one, but you can have it start printing at a later page. Enter the following command to resume the request at a different page:

lp -i ***request-id*** **-H resume -P** ***starting-page–***

The final dash is needed to specify the starting page and all subsequent pages.

NOTE The ability to print a subset of pages requires the presence of a filter that can handle this. The default filter used by the print service cannot handle it. An attempt to resume a request on a later page is rejected if an appropriate filter is not being used.

Moving a request to the head of the queue

You can move a print request to the head of the queue, where it is the next job eligible for printing. If it must start printing immediately, but another request is currently printing, you can hold the other request as described previously.

Enter the following command to move a print request to the head of the queue:

lp -i ***request-id*** **-H immediate**

Only the system administrator can move a request like this; regular users cannot use the **-H immediate** option.

NOTE If you set more than one request for immediate printing, they print in the reverse order set; that is, the request moved to the head of the queue most recently prints first.

Examining a printer configuration

Once you define a printer configuration, you probably want to review it to see if it is correct. If after examining the configuration you find you made a mistake, you can run the necessary command (or **sysadmsh** selection) for the characteristic concerned; you do not have to redefine the entire configuration.

Use the **lpstat** command to examine both the configuration and the current status of a printer. A short form of this command gives just the status; you can use it to see if the printer exists and if it is busy, idle, or disabled. A long form of the command adds the complete configuration. Enter one of the following commands to examine a printer:

lpstat -p ***printername***
lpstat -p ***printername*** **-l**

The second command is the long form. With either command you should see something like the following:

```
printer printer-name now printing request-id. enabled
since date.

printer printer-name is idle. enabled since date.

printer printer-name disabled since date.
        reason

printer printer-name waiting for auto-retry.
       reason
```

The "waiting for auto-retry" output shows that the print service failed in trying to use the printer (because of the ***reason*** shown) and that the print service will try again later.

With the long form of the command, you may also see the following items on the output:

```
Form mounted: form-name
Content types: content-type-list
Printer type: printer-type
Description: comment
Connection: connection-info
Interface: path-name
On fault: alert-method
After fault: fault-recovery
Users allowed:
        user-list
Forms allowed:
        form-list
Banner required
Character sets:
        character-set-list
Default pitch: integer CPI, integer LPI
Default page size: scaled-decimal-number wide,
   scaled-decimal-number long
Default port settings: stty-option-list
```

Using forms and filters

A preprinted form is a paper image of a blank form that you can 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.

The print service also permits the use of filters, which are used to accomplish three things:

- Convert a user's file into a data stream that prints properly on a given printer.
- Handle the various modes of printing that people may request with the **-y** option to the **lp** command, such as two-sided printing, landscape printing, draft or letter quality printing, and so on.
- Detect printer faults and inform the print service so that the latter can alert you.

There are very few cases where it will be necessary to use forms and filters, but these features are supported. For complete information, consult the **lpfilter**(ADM) and **lpforms**(ADM) manual pages.

Using the information in the request log

The directories */usr/spool/lp/temp* and *usr/spool/lp/requests* contain files that describe each request that has been submitted to the print service. Each request has two files, one in each directory, that contain information about the request. The information is split to put more sensitive information in the */usr/spool/lp/requests* directory where it can be kept secure. The request file in */usr/spool/lp/temp* is safe from all except the user who submitted the request, while the file in */usr/spool/lp/requests* is safe from even the submitting user.

These files remain in their directories only as long as the request is on the queue. Once the request is finished, the information in the files is combined and appended to the file */usr/spool/lp/logs/requests*.

The request log has a simple structure that makes it easy to extract data using common UNIX system shell commands. The requests are listed in the order in which they were printed and are separated by lines that give the request ID. Each line below the separator line is marked with a single letter that identifies the kind of information contained in the line. Each letter is separated from the data by a single space. Table 6-7 lists the log file codes.

Table 6-7 Request log entries

Letter	Content of line
=	This is the separator line, containing the request ID, the user and group IDs of the user, the total number of bytes in the original (unfiltered) files, and the time when the request was queued. These items are separated by commas and are in the order just named. The user ID, group ID, and sizes are preceded by the word **uid**, **gid**, and **size**, respectively.
C	The number of copies printed.
D	The printer or class destination or the word **any**.
F	The name of the file printed. This line is repeated for each file printed, and files are printed in the order given.
f	The name of the form used.
H	The type of special handling used, spelled out (**resume**, **hold**, **immediate**). The only useful value found in this line is **immediate**.
N	The type of alert used when the print request is successfully completed. The type is the letter **M** if the user was notified by mail, or **W** if the user was notified by a message to his or her terminal.

(Continued on next page)

Table 6-7 Request log entries
(Continued)

Letter	Content of line
O	The **-o** options.
P	The priority of the print request.
p	The list of pages printed.
r	This single letter line is present if the user asked for raw processing of the files (the **-r** option of the **lp** command.)
S	The character set or print wheel used.
s	The outcome of the request as a combination of individual bits expressed in hexadecimal form. While several bits are used internally by the Spooler, the most important bits are listed below: **0x0004** Slow filtering finished successfully. **0x0010** Printing finished successfully. **0x0040** The request was canceled. **0x0100** The request failed filtering or printing.
T	The title placed on the banner page.
t	The type of content found in the file(s).
U	The name of the user who submitted the print request.
x	The slow filter used for the request.
Y	The list of special modes to give to the filters used to print the request.
y	The fast filter used for the request.
z	The 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 request was moved to another destination by the print service administrator.

Customizing the print service

Although the print service tries to be flexible enough to handle most printers and printing needs, it cannot be complete. You may buy a printer that does not quite fit into the way the print service handles printers or may have a printing need that the standard features of the print service do not accommodate.

You can customize the print service in a few ways. This section tells you how you can:

- adjust the printer port characteristics
- adjust the *terminfo* database
- write an interface program

The diagram in Figure 6-1 gives an overview of the processing of a print request.

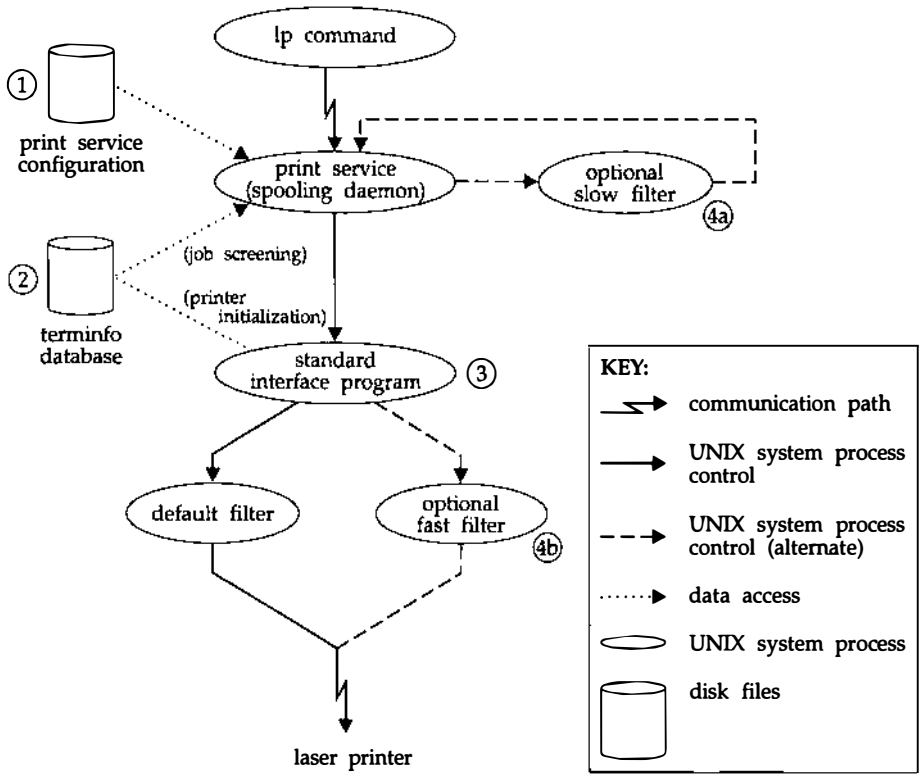


Figure 6-1 How the print service processes print request lp -d laser file

Each print request is sent to a *spooling daemon* (background program) that keeps track of all the requests. The daemon is created when you start the print service. This UNIX system process is also responsible for keeping track of the status of the printers and slow filters; when a printer finishes printing a user's file, the daemon starts it printing another request, if one is queued.

You can customize the print service by adjusting or replacing some of the pieces shown in Figure 6-1. (The numbers are keyed to the diagram.)

1. For most printers, you need only change the printer configuration stored on disk. The earlier sections of this chapter have explained how to do this. Some of the more printer-dependent configuration data are the printer port characteristics: baud rate, parity, and so on.

2. For printers that are not represented in the *terminfo* database, you can add a new entry that describes the capabilities of the printer. This database is used in two parallel capacities: screening print requests to ensure that those accepted can be handled by the desired printer and setting the printer so it is ready to print the request.

   For instance, if the *terminfo* database does not show a printer capable of setting a page length requested by a user, the spooling daemon rejects the request. On the other hand, if it does show it capable, then the same information is used by the interface program to initialize the printer.

3. For particularly difficult printers, or if you want to add features not provided by the delivered print service, you can change the standard interface program. This program is responsible for managing the printer: it prints the banner page, initializes the printer, and invokes a filter to send copies of the user's files to the printer.

4a,b. To provide a link between the applications used on your system and the printers, you can add slow and fast filters. Each type of filter can convert a file into another form, mapping one set of escape sequences into another, for instance, and can provide special setup by interpreting print modes requested by a user. Slow filters are run separately by the daemon to avoid tying up a printer. Fast filters are run so their output goes directly to the printer; thus, they can exert control over the printer.

Adjusting the printer port characteristics

You should make sure that the printer port characteristics set by the print service match the printer communication settings. The standard printer port settings were designed to work with typical UNIX files and many printers, but they do not work with all files and printers. This is not really a customizing step, because a standard feature of the print service is to allow you to specify the port settings for each printer. However, it is an important step in getting your printer to work with the print service, so it is described in more detail here.

When you add a new printer, read the documentation that comes with it so that you understand what it expects from the host (the print service). Then read the manual page for the **stty**(C) command. It summarizes the various characteristics that can be set on a terminal or printer port.

Only some of the characteristics listed in the **stty**(C) manual page are important for printers. The ones likely to be of interest to you are listed in the following table (but you should still consult the **stty**(C) manual page for others).

Printers connected directly to computers and those connected over some networks require that the printer port characteristics be set by the interface program. These characteristics define the low-level communications with the printer. Included are the baud rate; use of XON/XOFF flow control; 7, 8, or other bits per byte; style of parity; and output post-processing. The standard interface program uses the **stty** command to initialize the printer port, minimally setting the baud rate and a few other default characteristics.

The default characteristics applied by the standard interface program are listed in Table 6-8.

Table 6-8 Default stty options

Default	Meaning
9600	9600 baud rate
cs8	8-bit bytes
-cstopb	1 stop bit per byte
-parenb	no parity generation
ixon	enable XON/XOFF flow control
-ixany	allow only XON to restart output
opost	post-process data stream as listed below
-oluc	do not map lowercase to uppercase
onlcr	map linefeed into carriage-return/linefeed
-ocrnl	do not map carriage-return into linefeed
-nocr	output carriage-returns even at column 0
nl0	no delay after linefeeds
cr0	no delay after carriage-returns
tab0	no delay after tabs
bs0	no delay after backspaces
vt0	no delay after vertical tabs
ff0	no delay after formfeeds

You may find that the default characteristics are sufficient for your printers. However, printers vary enough that you are likely to find that you have to set different characteristics. See the **stty**(C) manual page to find the complete list of characteristics.

If you have a printer that requires printer port characteristics other than those handled by the **stty** program, you must customize the interface program.

When you add a new printer, you can specify an additional list of port characteristics that should be applied when printing each user's file. The list you give will be applied after the default list so that you do not need to include default items that you do not want to change in your list. Specify the additional list as follows:

/usr/lib/lpadmin -p *printer-name* -o "stty='*stty-option-list*'"

Δ **sysadmsh** users select: Printers ⇨ Configure ⇨ Parameters

Note that both the double quotes and single quotes are needed if you give more than one item in the ***stty-option-list***. If you do not include alternate printer port characteristics, the default list in the table will be used.

As one example, suppose your printer is to be used for printing graphical data, where linefeed characters should be output alone without an added carriage-return. You would enter the following command:

/usr/lib/lpadmin -p *printer-name* -o "stty=-onlcr"

Note that the single quotes are omitted because there is only one item in the list.

As another example, suppose your printer requires odd parity for data sent to it. You would enter the following command:

/usr/lib/lpadmin -p *printer-name* -o "stty='parenb parodd cs7'"

Adjusting the terminfo database

The print service relies on a standard interface and the *terminfo* database to initialize each printer and set up a selected page size, character pitch, line pitch, and character set. Thus, it is usually sufficient to have the correct entry in the *terminfo* database (*/usr/lib/terminfo/terminfo.lp*) to add a new printer to the print service. Several entries for popular printers are delivered in *terminfo* database entries with the print service package.

Each printer is identified in the *terminfo* database with a short name; this kind of name is identical to the kind of name used to set the **TERM** shell variable. For instance, the AT&T model 455 printer is identified by the name "455".

If you cannot find a *terminfo* entry for your printer, you should add one. If you do not, you may still be able to use the printer with the print service, but you cannot get automatic selection of page size, pitch, and character sets, and you may have trouble keeping the printer set in the correct modes for each print request. Another option to follow instead of updating the *terminfo* entry is to customize the interface program used with the printer. See the next section for details on how to do this.

There are hundreds of items that can be defined for each terminal or printer in the *terminfo* database. However, the print service uses less than fifty of these, and most printers need even less than that. Table 6-9 lists the items that need to be defined (as appropriate for the printer) to add a new printer to the print service.

Table 6-9 terminfo entry definitions for printers

terminfo item	Meaning
Booleans:	
daisy	printer needs operator to change character set
Numbers:	
bufsz	number of bytes buffered before printing
cols	number of columns in a line
it	tabs initially every # spaces
lines	number of lines on a page
orc	horizontal resolution in units per character
orhi	horizontal resolution in units per inch
orl	vertical resolution in units per line
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

(Continued on next page)

Table 6-9 terminfo entry definitions for printers
(Continued)

terminfo item	Meaning
iprog	pathname of initializing program
cud	move carriage down # lines
cuf	move carriage right # columns
rep	repeat a character # times
vpa	vertical position absolute
scs	select character set
smgb	set bottom margin at current line
smgbp	set bottom margin
smgl	set left margin at current column
smglp	set left margin
smgr	set right margin at current column
smgrp	set right margin
smgt	set top margin at current line
smgtp	set top margin
scsd	start definition of a character set
ht	tab to next 8-space tab stop

Consult the manual page for the **terminfo**(M) file structure for details on how to construct a *terminfo* database entry for a new printer.

Once you make the new entry, you need to compile it into the database using the **tic** program. Just enter the following command:

tic ***filename***

filename is the name of the file containing the *terminfo* entry you have crafted for the new printer.

NOTE The print service gains much efficiency by *caching* information from the *terminfo* database. If you add or delete *terminfo* entries or change the values that govern pitch settings, page width and length, or character sets, you should stop and restart the print service so it can read the new information.

How to write an interface program

If you have a printer that is not supported by simply adding an entry to the *terminfo* database, or if you have printing needs that are not supported by the standard interface program, you can furnish your own interface program. It is a good idea to start with the standard interface program (or one of the many others found in */usr/spool/lp/model*) and change it to fit, rather than starting from scratch. The "standard" script is found under the name */usr/spool/lp/model/standard*.

What does an interface program do?

Any interface program performs the following tasks:

- Initializes the printer port, if needed. The generic interface program uses the **stty** command to do this.
- Initializes the physical printer. The generic interface program uses the *terminfo* and the **TERM** shell variable to get the control sequences to do this.
- Prints a banner page, if needed.
- Prints the correct number of copies of the request content.

An interface program is not responsible for opening the printer port. This is done by the print service, which calls a dial-up printer if that is how the printer is connected. The printer port connection is given to the interface program as standard output, and the printer is set to be the controlling terminal for the interface program so that a hang-up of the port causes a **SIGHUP** signal to be sent to the interface program.

A customized interface program must not terminate the connection to the printer or in any fashion uninitialize the printer. This restriction allows the print service to use the interface program only for preparing the printer and printer port, while the printing of content is done elsewhere, by the print service, for example, for preprinted form alignment patterns.

How an interface program is used

When the print service routes an output request to a printer, the interface program for the printer is invoked as follows:

/usr/spool/lp/admins/lp/interfaces/*printer id user title copies options file1 file2 ...*

Arguments for the interface program are:

printer	Printer name. This is the name given to the interface program itself.
id	Request id returned by **lp**.
user	Login name of user who made the request.
title	Optional title specified by the user.
copies	Number of copies requested by user.
options	List of options separated by blanks, specified by user or set by the print service.
file	Full pathname of a file to be printed.

When the interface program is invoked, its standard input comes from */dev/null*, its standard output is directed to the printer port, and its standard error output is directed to a file that will be given to the user who submitted the print request.

The standard interface recognizes the following values in the list in ***options***:

nobanner This option is used to skip the printing of a banner page; without it, a banner page is printed.

nofilebreak
This option is used to skip page breaks between separate data files; without it, a page break is made between each file in the content of a print request.

cpi=*decimal-number1*
lpi=*decimal-number2*
These options say to print with ***decimal-number1*** characters per inch and ***decimal-number2*** lines per inch, respectively. The standard interface program extracts from the *terminfo* database the control sequences needed to initialize the printer to handle the character and line pitches.

The words **pica**, **elite**, and **compressed** are acceptable replacements for the ***decimal-number1*** and are synonyms for 10 characters per inch, 12 characters per inch, and as many characters per inch as possible.

length=*decimal-number1*
width=*decimal-number2*
These options specify the length and width, respectively, of the pages to be printed. The standard interface program extracts from the *terminfo* database the control sequences needed to initialize the printer to handle the page length and page width.

stty='*stty-option-list*'
The ***stty-option-list*** is applied after a default ***list*** as arguments to the **stty** command. The default list is used to establish a default port configuration; the additional list given to the interface program is used to change the configuration as needed.

The above options are either specified by the user when issuing a print request or by the print service from defaults given by the administrator for the printer (**cpi, lpi, length, width, stty**) or for the preprinted form used in the request (**cpi, lpi, length, width**).

Additional printer configuration information is passed to the interface program in shell variables:

TERM=*printer-type*
: This shell variable specifies the type of printer. The value is used as a key for getting printer capability information from the extended *terminfo* database.

FILTER='*pipeline*'
: This shell variable specifies the filter to use to send the request content to the printer; the filter is given control of the printer.

CHARSET=*character-set*
: This shell variable specifies the character set to be used when printing the content of a print request. The standard interface program extracts from the *terminfo* database the control sequences needed to select the character set.

A customized interface program should either ignore these options and shell variables or should recognize them and treat them in a consistent manner.

Customizing the interface program

Make sure that the custom interface program sets the proper **stty** modes (terminal characteristics such as baud rate or output options). The standard interface program does this, and you can follow suit. Look for the section that begins with the shell comment:

```
## Initialize the printer port
```

Follow the code used in the standard interface program. It sets both the default modes and the adjusted modes given by the print service or the user with a line like the following:

stty *mode options* 0<&1

This command line takes the standard input for the **stty** command from the printer port. An example of an **stty** command line that sets the baud rate at 1200 and sets some of the option modes is shown here:

stty -parenb -parodd 1200 cs8 cread clocal ixon 0<&1

One printer port characteristic not set by the standard interface program is hardware flow control. This is set depending on your computer hardware. The code for the standard interface program suggests where this and other printer port characteristics can be set. Look for the section that begins with the shell comment:

```
# Here you may want to add other port initialization code.
```

Because different printers have different numbers of columns, make sure the header and trailer for your interface program correspond to your printer. The standard interface program prints a banner that fits on an 80-column page (except for the user's title which may be longer). Look for the section in the code for the standard interface program that begins with the shell comment:

```
## Print the banner page
```

The custom interface program should print all user-related error messages on the standard output or on the standard error. The messages sent to the standard error are mailed to the user; the messages printed on the standard output end up on the printed page, where they can be read by the user when they pick up the output.

When printing is complete, your interface program should exit with a code that tells the status of the print job. Exit codes are interpreted by the print service as shown in Table 6-10.

Table 6-10 Exit codes

Code	Meaning to the print service
0	The print request completed successfully. If a printer fault occurred, it was cleared.
1 to 127	A problem was encountered in printing this particular request (for example, too many non-printable characters or the request exceeds the printer capabilities). This problem does not affect future print requests. The print service notifies the person who submitted the request (via **write**(C) or **mail**(C)) that there was an error in printing it. If a printer fault occurred, it was cleared.
128	Reserved for internal use by the print service. Interface programs must not exit with this code.
129	A printer fault was encountered in printing the request. This problem affects future print requests. If the fault recovery for the printer directs the print service to wait for the administrator to fix the problem, it disables the printer. If the fault recovery is to continue printing, the print service does not disable the printer but tries printing again in a few minutes.
> 129	These codes are reserved for internal use by the print service. Interface programs must not exit with codes in this range.

As the table shows, one way of alerting the administrator to a printer fault is to exit with a code of 129. Unfortunately, if the interface program exits, the print service has no choice but to reprint the request from the beginning when the fault is cleared. Another way of getting an alert to the administrator but without requiring reprinting the entire request, is to have the interface program send a fault message to the print service but wait for the fault to clear. When the fault clears, the interface program can resume printing the user's file. When finished printing, it can give a zero exit code as if the fault never occurred. An added advantage is that the interface program can detect when the fault is cleared automatically so that the administrator does not have to enable the printer.

Fault messages can be sent to the print service using the **lp.tell** program. This is referenced using the **LPTELL** shell variable in the standard interface code. The program takes its standard input and sends it to the print service, where it is put into the message that alerts the administrator to the printer fault. If its standard input is empty, **lp.tell** does not initiate an alert. Examine the standard interface code immediately after these comments for an example of how the **lp.tell** (**LPTELL**) program is used:

```
# Here's where we set up the $LPTELL program to capture
# fault messages.

# Here's where we print the file.
```

With the special exit code 129 or the **lp.tell** program, there is no longer the need for the interface program to disable the printer itself. Your interface program can disable the printer directly, but doing so overrides the fault alerting mechanism. Alerts are sent only if the print service detects the printer has faulted, and the special exit code and the **lp.tell** program are its main detection tools.

If the print service has to interrupt the printing of a file at any time, it kills the interface program with a signal 15 (see **signal**(S) in the *Programmer's Reference*, and **kill**(C) for more information).

If the interface program dies from receipt of any other signal, the print service assumes that future print requests are not affected and continues to use the printer. The print service notifies the person who submitted the request that it did not finish successfully.

The signals **SIGHUP**, **SIGINT**, **SIGQUI**, and **SIGPIP** (trap numbers 1, 2, 3, and 13) start out being ignored when the interface is invoked. The standard interface changes this to trap these signals at appropriate times. The standard interface considers receipt of these signals as meaning the printer has a problem and issues a fault. This is the program the print service uses to manage the printer each time a file is printed. It has four main tasks:

- to initialize the printer port (the connection between the computer and the printer),
- to initialize the printer (restore it to a normal state in case a previously printed file has left it in an unusual state) and set the character pitch, line pitch, page size, and character set requested by the user,
- to print a banner page, and
- to run a filter to print the file.

How to add an interface program

If you do not choose an interface program, the standard one provided with the print service is used. This should be sufficient for most of your printing needs. If you prefer, however, you can change it to suit your needs or completely rewrite your own interface program, and then specify it when you add a new printer.

If you plan to use the standard interface program, you need not specify it when adding a printer. However, if you use a different interface program, you can either refer to it by its full pathname or by another printer using the same interface program.

To identify a customized interface program by name, give the printer name and the pathname of the interface program as follows:

/usr/lib/lpadmin -p ***printername*** **-i** ***pathname***

To identify a customized interface program by reference to another printer, give the printer names as follows:

/usr/lib/lpadmin -p ***printername1*** **-e** ***printername2***

printername1 should be replaced with the name of the printer you are adding; ***printername2*** should be replaced with the name of the printer already added that is using the customized interface program.

To identify an interface program by reference to a model interface program, give the printer name and model name as follows:

/usr/lib/lpadmin -p ***printername*** **-m** ***modelname***

This takes the interface program from the file */usr/spool/lp/model/modelname.*

NOTE If ***printername*** is a new printer to be added, the **lpadmin** command line must include the **-u** (dial-out) or **-v** (device) option. The printer must then be enabled and accepted.

Configuring a network printer

To configure the spooler to print to a printer on another machine accessed via network (such as TCP/IP or UUCP) you must set up the printer as follows:

1. Choose Printers ➪ Configure ➪ Add from **sysadmsh**. Give the printer a name, and choose the existing interface script *network* (or *network.ps* for a PostScript-capable printer). The Connection should be "Direct" and the Device should be "Hardwired." If you are adding a printer on a TCP/IP network, use */dev/null* for the device name, otherwise, this should be left blank.
2. Create a file called */usr/spool/lp/remote*. This file should contain the following line:

   name:⟨Tab⟩***command***

   where ***name*** is the name of the printer on this machine and ***command*** is the command which will transfer the file to be printed to the remote system. For example, if the name you gave the printer is *remprint* and the name of the remote machine to which this printer is connected is *wally* and the name of the printer on machine *wally* is called *beaver* then the line would read:

   For UUCP:

   ```
   remprint:       uux - wally!/usr/bin/lpr -dbeaver
   ```

   For TCP/IP:

   ```
   remprint:       /usr/spool/lp/bin/rlpcmd wally lp -dbeaver
   ```

3. The file */usr/spool/lp/remote* should have owner and group *lp*, and be publicly readable. To set this, issue the following commands:

   chown lp /usr/spool/lp/remote
   chgrp lp /usr/spool/lp/remote
   chmod 444 /usr/spool/lp/remote

4. For a TCP/IP network, a user need not have an account on the print server; all that is necessary is consistency among the *lp* accounts on the network. The *.rhosts* file should contain entries in the following format:

   client `lp`

   All print jobs are submitted under the name *lp*.

NOTE If you have configured a remote printer over UUCP, some information about dialup printers also applies to UUCP. See the next section about dialup printers.

Configuring a dialup printer

To set up a dialup printer, you must do the following:

1. Because the **cu** program accesses a printer in the same way the print service does, you should set up the files as though preparing access to the printer for **cu**. The **cu** command is not used to access printers but can serve as a yardstick when setting up files: if **cu** can access a printer, the print service will be able to access it, too. (See the "Using modems" chapter in this guide for details about setting up a modem connection.)
2. Make certain the following line is present in */etc/default/lpd*:

   `DIALUPPRINTER=YES`

   If it is missing or the entry reads "NO", change it.
3. Choose Printers ⇨ Configure ⇨ Add from **sysadmsh**. Complete the fields as follows:

Name:	dialup (or any name)
Interface:	dumb
Dial-up information:	the phone number of the remote system
Device:	hardwired

   Press ⟨Ctrl⟩x when you are finished.
4. Enter the following command to ensure failure info is mailed to *root*:

   /usr/lib/lpadmin -p dialup -A mail
5. You must then prepare the print system to accept jobs and enable the device:

   accept dialup
   enable dialup
6. You should then be able to submit job with the following command:

   lp -ddialup ***filename***

The following applies to dialup and UUCP network printers: If the printer or port is busy, the print service will automatically retry later. This retry rate is 10 minutes if the printer is busy and 20 minutes if the port is busy. The rate is not adjustable. However, you can force an immediate retry by issuing an **enable** command for the printer. If the port or printer is likely to be busy for an extended period, you should issue a **disable** command.

The **lpstat -p** command reports the reason for a failed dial attempt. Also, if you are alerted to a dialing fault the alert message will give the reason for the fault. (Refer to "Fault alerting" (page 131) for more information on fault alerting.) These messages are identical to the error messages produced by the UUCP system for similar problems. Refer to "UUCP STATUS error messages" (page 372) for an explanation of the reasons for failure.

Using a printer without the spooler

If you use a printer without the spooler, any **stty** settings you have specified for use with that printer do not stay in effect. The spooler opens the file and then runs the **stty** commands as specified in the printer interface script. To use a printer without the spooler, follow the instructions in this section.

While logged in as *root*, give the following commands or insert them into the initialization file */etc/rc2.d/S80lp* before the line that calls */usr/lib/lpsched*. The first command is for serial printers, and 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* &

where ***baud*** is the baud rate of the printer, **tty*n*** or **lp*n*** is the serial or parallel device name, 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, make sure you kill this process first.

NOTE With certain multiport cards, it is necessary to add a **sleep** command after the initialization program supplied with the card, ***initprogram***, followed by the **stty holdopen** command:

***initprogram* &**
sleep 3

Creating 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 the device is opened after the system starts up. This is done on the first open only, so printers with large text buffers are not flushed by the sending of another file.

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 ribbons). The symptom of this situation is that the printer works fine until it is turned off and then back on.

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 in use. This init device file can be used when necessary to initialize the printer.

1. Log in as super user.
2. Determine which device is the parallel port you are using. This example assumes the device is the main parallel port (*/dev/lp0*).

3. Run the command:

   cd / ; fixperm -csdLPR /usr/lib/mkdev/perms/LPR

   to correct/initialize file permissions for the printer devices file */usr/lib/mkdev/perms/LPR*.

4. Use the **sysadmsh** selection:

   Printers ⇨ Configure ⇨ Modify

   to associate one of the parallel **init** devices (*lp0i*, *lp1i*, *lp2i*) with the printer.

   If your printer needs an **init** when it is turned off and on, use the following command line after the printer is turned on. Initialize the printer before the first file is sent to the printer (this example assumes the main parallel port):

   > /dev/lp0i

   If your printer needs 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 will then send an **init** every time a file is sent to the printer.

Chapter 7

Using floppy disks and tape drives

An important part of any computer system is the ability to offload files and restore them when needed. There are several types of media used to store and recall files. Among these are floppy disks and magnetic tape devices. This chapter explains how to use storage media with your system. Your system should come with at least a floppy disk drive already installed and ready to run. For information on configuring tape drives, consult the *Hardware Configuration Guide.*

Using tape drives

A tape drive is a mass storage device that uses magnetic tape cartridges to store data. A tape can hold many times the data that can be stored on floppies, making it much more useful for large backup operations.

The drives that are supported are listed in the SCO list of compatible hardware. For hardware-specific information, refer to the hardware manual for your drive, the manufacturer of the drive, and the **tape**(HW) manual page.

Tape device files

After the drive is configured, a series of device files are created to access the tape drive. Some are linked to the basic tape device file, */dev/rct0*. Others include the "no-rewind" device, which does not rewind the tape after access, and the "no-unload" device, which does not eject the tape after use. Table 7-1 lists the prefixes used in QIC-02 and SCSI tape device names.

Table 7-1 Tape device prefixes

Prefix	Meaning	Example
e	ECC device (QIC-02 only)	/dev/erct0
rh	high density (9-track only)	/dev/rhStp0
nr	no rewind, no unload	/dev/nrStp0
nur	rewind, no unload	/dev/nurStp0
ur	rewind, unload	/dev/urStp0
r	rewind on close	/dev/rStp0, /dev/rct0
x	override/control	/dev/xStp0, /dev/xct0, /dev/xft0, /dev/xctmini
n	no rewind	/dev/nrct0

Table 7-2 lists the main devices created for tape drives and the default device to which they are linked.

Table 7-2 Tape device files

Tape device	Device file	Default device
QIC-02	/dev/rct0	/dev/rct0
Mini (irwin)	/dev/rmc0	/dev/rctmini
QIC-40/80	/dev/rft0	/dev/rctmini
SCSI cartridge	/dev/urStp0	/dev/rct0
DAT/Exabyte	/dev/nurStp0*	/dev/rct0
9-track	/dev/nurStp0	/dev/rct0
Compaq	/dev/rmt/cst2*	/dev/rct0

* Some drives have additional device files that depend on brand and model. For example, certain DAT drives support tape partitioning; see **dat**(HW) and **tape**(HW).

In addition, some devices are linked to equivalent standard UNIX devices in the */dev/rmt/** hierarchy.

Changing the default tape drive

If you install more than one tape drive, you are prompted by **mkdev tape** for the drive you want linked to the default tape device files. For example, if you have a both a SCSI cartridge tape drive and an 8-mm tape drive installed, only one of them can be linked to the default device */dev/rct0*. If you decide later to change the drive linked to the default device, you can do so with option 3 of the **mkdev tape** menu.

Setting the default tar(C) device

After you install your tape drive, you must enter the correct size setting in the */etc/default/tar* file. When you edit the file, you see several entries for various default devices. Figure 7-1 shows the */etc/default/tar* file provided with the 96tpi distribution. Note that the sizes for */dev/rct0* and */dev/rctmini* are 0, meaning there is no size associated with this device. This is acceptable if you never use an entire tape. However, if you exceed the tape's capacity, **tar** cannot create a multi-volume archive properly. If you plan to create archives that span tape volumes, you must add the proper tape size as described in this section.


```
#device                    block   size    tape
archive0=/dev/rfd048ds9    18      360     n
archive1=/dev/rfd148ds9    18      360     n
archive2=/dev/rfd096ds15   10      1200    n
archive3=/dev/rfd196ds15   10      1200    n
archive4=/dev/rfd096ds9    18      720     n
archive5=/dev/rfd196ds9    18      720     n
archive6=/dev/rfd0135ds18  18      1440    n
archive7=/dev/rfd1135ds18  18      1440    n
archive8=/dev/rct0         20      0       y
archive9=/dev/rctmini      20      0       y
# The default device...
archive=/dev/rfd096ds15    10      1200    n
```

Figure 7-1 /etc/default/tar file

QIC cartridge drives

If a QIC-02 drive is your default tape drive, it uses the */dev/rct0* entry. The cartridge sizes are indicated in Table 7-3. You can edit */etc/default/tar* and set the appropriate size, or create multiple entries to accommodate different tape sizes.

Table 7-3 QIC cartridge sizes

Length in feet	Entry in "Size" field	Capacity in megabtyes
300	30000	30
450	45000	45
600	60000	60
1200	120000	120
1500	150000	150
2500	250000	250
5250	525000	525

Mini-cartridge (Irwin) drives

Find the entry in your */etc/default/tar* file for */dev/rctmini*. In Figure 7-1, this is **archive9**. The correct size for your *rctmini* device varies with the size of the tape you use. See Table 7-4.

Table 7-4 Mini-cartridge sizes

Tape size in megabytes	Actual capacity in megabytes	Entry in "Size" field
40	35	35000
80	72	72000
120	118	118000

The actual size for Irwin tapes is smaller because some storage is reserved for error correction code (ECC) data.

DAT and 8-mm (Exabyte) drives

If you wish to use the default file with a DAT or 8-mm drive and it is not the default tape device, you must change one of the entries to use the */dev/rStp* device:

```
archive9=/dev/rStpn     20      size     y
```

Replace ***n*** with the number or your drive and ***size*** with the size (in Kbytes) of the cartridges you are using.

Archiving files on tape

You use a tape drive much like a floppy, but the volume of data stored is much greater. Tapes are much better for storing (backing up) entire filesystems. The **tar**(C) command is the recommended archive program for users and is best used for general archiving or transporting of files. Other programs such as **cpio**(C) are meant for system administrators making copies of entire filesystems. Consult "Backing up filesystems" in this guide for making regular backups of filesystems.

The **cpio**(C) command is a general-purpose archive program that uses a different format than **tar**. The **dd**(C) program transfers or converts archives of unusual format; the input and output format can be specified on the command line.

The tar command

The **tar** command is useful for making a backup copy of entire directories. The command has the syntax:

tar cvf ***devicefile files***

The *devicefile* is the filename that corresponds to the cartridge drive. *files* are the names of the files or directories to be copied. For example, to copy all the files in the directory */u/bogart* to the cartridge drive */dev/rct0*, enter:

tar cvf /dev/rct0 /u/bogart

Δ **sysadmsh** users select: Media ⇨ Archive

WARNING **tar** will not accept pathnames longer than 100 characters. It will fail with the following error message:

```
tar: filename : pathname too long
```

cpio(C) accepts pathnames of 1024 characters.

Alternatively, you can use the shorter */etc/default/tar* entry (**archive8**) shown in Figure 7-2 rather than entering the full device name:

tar cv8 /u/bogart

To restore files stored on tape, insert the cartridge containing the files or directories you wish to restore and enter the following command:

tar xvf ***devicefile***

Δ **sysadmsh** users select: Media ⇨ Extract

Again, you can also use the shorter form:

tar xv8

NOTE When creating a **tar** archive, you need to be conscious of absolute and relative pathnames. If you specify an *absolute* pathname when creating an archive (for example: **tar cv2 /u/wanda/tempfile**), then it will be extracted with that same pathname (and thus appear below the current directory). If you do not wish this to occur, use *relative* pathnames when creating an archive (for example: **tar cv2 tempfile**).

You can list (or verify) a **tar** archive by asking for a listing with the **t** option:

tar tv2

Tape drive maintenance

The **tape**(C) utility performs various tape maintenance operations on all tape drives. **tape** sends commands and receives status from the tape drive. The basic form of the command is:

tape ***command*** [***devicefile***]

For example, to rewind a cartridge tape device, enter:

tape rewind

When invoked without a device, the default device (in */etc/default/tape*) is used.

Other commands include the following:

erase erases tape cartridge and re-tensions

reset resets tape controller and tape drive, clears error conditions, and returns tape subsystem to power-up state

reten re-tensions tape cartridge. Should be used periodically to remedy slack tape problems that generate an unusually large number of tape errors or, in extreme cases, actually tangle the tape in the drive.

Refer to the **tape**(C) manual page for more information on these and other commands.

After certain tape operations are executed, the system returns a prompt before the tape controller has finished its operation. If you enter another tape command too quickly, the message "device busy" is displayed until the tape device is finished with its previous operation.

You should clean the tape drive heads and re-tension cartridges to keep it operating error-free.

Irwin drive commands

There are certain commands that are specific to Irwin drives:

info prints information about the inserted cartridge, including whether or not it is formatted, what type of format, and write protect status

capacity or **kapacity**
displays the capacity of the inserted cartridge in 512- and 1024-byte blocks, respectively

Tape driver error correction code (ECC) support

Tape ECC is supported on QIC-02, mini-cartridge (Irwin), and QIC-40/80 drives. ECC is not available on SCSI tape drives.

QIC-02 ECC

The QIC-02 ECC tape device node, */dev/erct0*, is automatically created when you run **mkdev tape**. To use ECC on the QIC-02, you must read and write from this device, not the normal */dev/rct0*. With tape drives that support cartridges larger than 60 Mbytes, it is a good idea to edit the */etc/default/tar* file and substitute */dev/erct0* for the normal tape device.

The error recovery scheme is 2/64, which means that two 512-byte blocks out of every 64 blocks can go bad and the driver corrects them. The probability of error with ECC is $1{:}10^{14}$. Standard drives have an error probability of $1{:}10^{9}$.

Be sure and label tapes that are created with the ECC device; these tapes cannot be read by standard devices. In addition, if transporting data from one machine to another, it is advisable to use the ECC device only if the target machine supports the ECC scheme.

Mini-cartridge (Irwin)

ECC encoding and decoding is automatic on the mini-cartridge (Irwin) 80-Mbyte drives; no ECC device is necessary. For every 29K written to the tape, 3K of ECC data is written. On smaller capacity mini-cartridge drives, for every 16K written to the tape, 2K of ECC data is written.

Using floppy disks

Floppy disks are the most convenient form of storage media. Depending on your floppy disk drive, you may be able to store from 360 Kbytes to 1.4 Mbytes on a single disk. Floppy disks can be used for simple data storage in **tar**, **cpio**, or **dd** formats or you can make a mountable filesystem on a floppy disk. The following sections explain how to use floppies for data storage and as extra filesystem space.

Formatting floppy disks

Floppy disks must be formatted before they can be used. The command to format a floppy disk is:

format /dev/*floppy-device*

Δ **sysadmsh** users select: Media ➪ Format

The floppy device you specify in the command relates to the type of disk drive and floppy you are using. For example, if you have a high-density 5.25-inch floppy disk drive, you can use it in high-density mode (96 tracks per inch) or in low-density mode (48 tpi). If you have high-density floppies to use with your drive, the floppy device to specify is:

/dev/rfd096

In this example, "rfd" indicates the raw floppy device, "0" indicates that this is the primary floppy drive, and "96" indicates high-density mode.

NOTE You must always specify the raw device; you cannot format the block device.

Similarly, if you wish to use low-density floppies and the low-density mode of the floppy drive, the device name is:

/dev/rfd048

In this example, "48" indicates the low-density mode of floppy drive "0".

/etc/default/format file

You can also define a default format device by adding an entry to the file */etc/default/format*. For example:

```
DEVICE=/dev/rfd096ds15
```

After adding the above line, you no longer have to specify the device name. In addition, it is possible to define that all floppies be verified, which confirms that the data on the floppy is readable. (This can also be specified on the command line with the **-v** option.) Automatic verification can be specified by the following entry:

```
VERIFY=Y
```

If this entry is placed in */etc/default/format*, all floppies formatted with the **format** command are verified. (To override verification, use the **-n** option on the command line.)

Refer to the **format**(C) manual page for more details.

Copying floppy disks

To ensure against the loss of data stored on floppy disks, any user can use the **diskcp**(C) command, or the **dd**(C) command to make copies of floppy disks on new, formatted disks.

diskcp makes use of **dd** and provides a simple interface to that program. **dd** is very powerful, and you can use it to perform many different kinds of copying.

If you need to copy one type of floppy to another (for example: 5¼ to 3½) you must use **dd** directly.

You must copy information onto formatted disks. If you format floppies, you can use them over again without reformatting.

NOTE If you have disks that were formatted under another operating system, you must reformat them before you can use them to make copies of UNIX disks. Be aware that floppies formatted under some operating systems cannot be used under other operating systems, even with reformatting.

You can use the **format** command to format floppies. Refer to "Formatting floppy disks" (page 181) for more infomation. The **diskcp** command can also format floppies for you.

To copy a floppy disk using **diskcp**, do the following:

Δ **sysadmsh** users select: Media ⇨ Duplicate

1. Insert the disk you want to copy, known as the *source* floppy, in drive 0, your primary floppy drive.
2. Insert another floppy in the other drive. This floppy is known as the *target* disk. Note that any information already on the target disk is destroyed.

   If you have only one disk drive, leave the source floppy in the drive. **diskcp** prompts you to remove the source disk at the correct time.
3. To format the floppy disk before the image is copied, enter the command:

   diskcp -f

   If your computer has two identical floppy drives, enter the following command to copy the image directly on the target floppy:

   diskcp -d

   If you do not need to format the target floppy, simply enter:

   diskcp
4. Follow the instructions as they appear on your screen. Note that, with a single drive system, you are prompted to remove the source disk and insert the target disk.

To copy a disk using **dd**, follow these steps:

1. Insert the disk to be copied into floppy drive 0.
2. Insert a formatted disk into drive 1. If necessary, you can format a disk with the **format** command. Refer to "Formatting floppy disks" (page 181) for more infomation.
3. Enter:

   dd if=/dev/fd0 of=/dev/fd1 count=*blkcount*

   The ***blkcount*** is the number of blocks on the disk to be copied. If you do not know this number, leave the **count=*blkcount*** section out of the command.

This command copies the first disk to the second, then displays a record of the number of blocks copied.

Using floppies for file storage

To use a floppy for simple file storage, first make sure that the floppy is formatted. Then, place the floppy in the floppy drive. You can use any of the standard UNIX file archiving utilities with floppy disks. These include **tar**, **cpio**, or **dd** formats.

tar is recommended for most file-archiving tasks. See "The tar command" (page 178) for details. For information on **cpio**, **dd**, and **backup** formats, see the associated manual pages.

Making filesystems on floppy disks

You can make a filesystem on a floppy disk similar to how you make one on a hard disk. Filesystems on floppy disks are portable and can be mounted on any UNIX system. A special directory called */mnt* is used for mounting filesystems that do not have a specified mounting point. Note that for system security, you must be logged in as *root* to use floppy filesystems, and only *root* can mount a floppy filesystem.

To make a portable filesystem on a floppy disk, use the following procedure:

1. Log in as *root* and enter the command:

   mkdev fd

   Δ **sysadmsh** users select: Filesystems ⇨ Floppy

2. You see the following menu:

```
Floppy Disk Filesystem Creation Program

       Choices for type of floppy filesystem.

       1. 48tpi, double sided, 9 sectors per track
       2. 96tpi, double sided, 15 sectors per track
       3. 135tpi, double sided, 9 sectors per track
       4. 135tpi, double sided, 18 sectors per track

Enter an option or q to quit:
```

   Enter the number of the disk type desired and press ⟨Return⟩.

3. If you have more than one floppy drive you see the following prompt:

```
Do you want to use floppy drive 0 or floppy drive 1
```

   Enter the number of the drive required and press ⟨Return⟩.

4. Next you see:

```
Choices for contents of floppy filesystem.

1. Filesystem
2. Bootable only        (96ds15 and 135ds18 only)
3. Root filesystem only (96ds15 and 135ds18 only)

Enter an option or enter q to quit:
```

Enter 1 and press ⟨Return⟩.

5. You see the following prompt:

```
Insert a type floppy into drive 0.
Press Return to continue or enter q to quit:
```

Press ⟨Return⟩.

6. The following prompt is displayed:

```
Would you like to format the floppy first? (y/n)
```

If you have already formatted the floppy, enter "n" and the filesystem is immediately created. If the floppy has not yet been formatted, enter "y" and you see:

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

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

7. Next you see a prompt for filesystem type:

```
Do you want to use the default file system type AFS (y/n)?
```

If you respond "n", you are asked to supply the type:

```
Please enter a file system type from <AFS, EAFS, S51K, XENIX>
or enter q to quit:
```

8. Enter the filesystem type and press ⟨Return⟩. When the process is complete, you see:

```
Filesystem creation complete.
```

9. Next you see this menu again:

```
Choices for contents of floppy filesystem.

1. Filesystem
2. Bootable only        (96ds15 and 135ds18 only)
3. Root filesystem only (96ds15 and 135ds18 only)

Enter an option or enter q to quit:
```

Now enter "q" and press ⟨Return⟩ to quit. Your floppy now contains a filesystem.

Mounting a floppy filesystem

To use a floppy filesystem, you must mount it on your system. For example, a 96-tpi floppy would use the following command:

mount /dev/fd096 /mnt

Δ **sysadmsh** users select: Filesystems ➪ Mount

Note that you use the floppy device *fd096* and not *rfd096*. When you mount a floppy filesystem, you must use the name without the preceding "r". As another example, if you choose to mount a filesystem on a 48-tpi disk, use the following command:

mount /dev/fd048 /mnt

When you give the **mount** command, the shell should return a prompt. This indicates that the filesystem was successfully mounted. You can now use the **cd** command to move into the filesystem and create files there as you would on any other filesystem:

cd /dev/fd048

When you are done and you wish to remove the floppy, be sure and **cd** out of the floppy filesystem and give the following command:

umount /dev/fd048

Δ **sysadmsh** users select: Filesystems ➪ Unmount

Your filesystem is immediately unmounted. Your files are contained on the floppy and can be stored or transported easily.

Chapter 8

Using MS-DOS and other DOS operating systems

Many users received MS-DOS, or other closely compatible DOS operating systems with their computer. This chapter explains how you can still use DOS utilities, files, and applications after you install the UNIX system. You can even access DOS files and directories on your UNIX system, or mount DOS filesystems and access the files directly. The UNIX system provides this facility so that you do not need to throw away your investment in DOS software, or buy another computer just to run a UNIX system.

Several programs make this coexistence possible. The **dos**(C) utilities allow access to DOS files on diskettes or on the DOS partition on the hard disk (provided the partition is unmounted). These utilities are discussed later in this chapter. The utility that partitions the disk is called **fdisk**(ADM) and is available in DOS and UNIX system versions. The next section explains how to use **fdisk** to create a DOS partition and a UNIX system partition on the same hard disk. Another section discusses installing a UNIX system partition on the hard disk along with DOS. There is also a section explaining various booting configurations, for users who mostly use the UNIX system and for users who mostly use DOS.

OS/2 coexistence

Although it may install successfully, OS/2 may not be bootable on your machine, regardless of whether a UNIX system partition is present or not; we cannot guarantee that OS/2 will work with your UNIX system. Refer to your computer's hardware documentation to determine if your machine is supposed to run OS/2. If you wish to use OS/2 and/or DOS on the same disk with your UNIX system, you must install them on the disk in the following order:

1. DOS Primary Partition
2. DOS Extended Partition
3. UNIX system
4. OS/2

NOTE If a second disk is installed you get a primary partition on the second disk and optionally an extended partition. You cannot boot DOS from the second extended partition.

There are no OS/2 tools available in UNIX System V (such as the DOS utilities described in this chapter). In addition, you must use **fdisk**(ADM) to switch to or from OS/2.

UNIX system **fdisk**(ADM) displays an OS/2 partition as DOS.

Partitioning the hard disk using fdisk

Each version of **fdisk** is documented in the respective operating system's manual. Unless otherwise noted, this chapter refers to the UNIX system version of **fdisk**(ADM).

fdisk is interactive, and uses a menu to display your options. Here is the main **fdisk** menu:

```
   1. Display Partition Table
   2. Use Entire Disk For UNIX
   3. Use Rest of Disk for UNIX
   4. Create UNIX Partition
   5. Activate Partition
   6. Delete Partition

Enter your choice or 'q' to quit:
```

You can do the following using **fdisk**:

- Set up separate areas (partitions) on your hard disk for different operating systems. The hard disk is divided into *tracks*. The number of tracks depends upon the size of the hard disk. A *partition* consists of a group of tracks. Each hard disk can accomodate four partitions.
- Specify one disk partition as "active". This means that when you turn on (boot) your computer, the operating system installed in the active partition will start running. The UNIX system partition must be active when you intend to use your UNIX system.
- Specify the number of tracks assigned to each partition. The number of available tracks will vary according to the size of your hard disk. Consult your *Installation Guide* for the recommended UNIX system partition size. The size of the UNIX system partition also depends on the number of software packages you want to install. You can install the UNIX system in this space, and have the rest of the space for user files and other software packages. Refer to the **custom**(ADM) manual page for information on how to install and remove software.
- Specify precisely where the partition begins and ends. **fdisk** will not allow you to construct overlapping partitions. You do not need to install your UNIX system in the first partition.

  You should start your DOS partition at the beginning of the disk, starting at cylinder 0 or cylinder 1.

  If you install a UNIX system partition on the same disk after the DOS, or extended DOS partition, start the UNIX system partition at the beginning of the next cylinder on the disk. To find the beginning of the next cylinder, note the ending track number of your DOS partition and start the UNIX system partition on the next track number that is a multiple of the number of heads on your hard disk. For example, if you have five heads on your hard disk and your DOS partition ends at track 103, start your UNIX system partition at track 105.

When you are running your UNIX system, the device name of the UNIX system partition is */dev/hd0a*. For more information about hard disk device names, see the **hd**(HW) manual page.

Displaying current partition table

One option of **fdisk** tabulates the current state of the partitions (the Display Partition Table option). This option lists, for each partition, whether the partition is active, the first track, the last track, the number of tracks used, and the associated operating system. If you enter the Display Partition Table option and press ⟨Return⟩ to see the partition table, the result will be similar to Example 8-1.

Example 8-1 Sample fdisk table

```
Current Hard Disk Drive: /dev/rhd00

+-------------+----------+-----------+---------+---------+---------+
| Partition   | Status   | Type      |   Start |     End |    Size |
+-------------+----------+-----------+---------+---------+---------+
| 1           | Inactive | DOS       |       5 |     398 |     393 |
| 2           | Inactive | DOS (ext) |     400 |    1219 |     819 |
| 3           | Active   | UNIX      |    1220 |    2220 |    1000 |
+-------------+----------+-----------+---------+---------+---------+

Total disk size: 2229 tracks (9 reserved for masterboot and diagnostics)
```

Switching operating systems

There are three ways to switch to the primary DOS partition once you have set up separate DOS and UNIX system partitions:

- enter **dos** at the boot prompt;
- use a floppy diskette that contains the files necessary to boot the DOS operating system, or
- use **fdisk** to change the current active partition.

When you use the boot prompt or a floppy to boot DOS, the UNIX system partition remains active even though you have switched operating systems. When you use **fdisk**, the UNIX system partition is inactive until you switch back to it.

To use the boot prompt method, enter:

dos

at the boot prompt:

```
SCO System V/386

Boot
: dos
```

NOTE The system boots from the first DOS partition found.

To use a floppy diskette to boot DOS, follow this procedure:

1. Make sure all users are logged off the system.
2. Run **shutdown**(ADM) to shut down the UNIX system. This command makes sure all users know the system is being shut down, terminates all processes, then halts the system.

3. Once the UNIX system has shut down, insert the bootable DOS diskette into the primary (boot) drive.
4. Boot DOS.
5. To get back to the UNIX system partition, remove any disks from the floppy drive(s) and press ⟨Ctrl⟩⟨Alt⟩⟨Del⟩, or the reset key, or turn the computer off, then on. Because the UNIX partition is still active, your UNIX system boots.

Remember that if you have an active UNIX system partition and boot DOS from a floppy you can transfer to C: to work with the DOS files.

The other way to change operating systems is to **run fdisk** and change the active partition from the UNIX system partition to DOS. Then, after you shut down the system (see the previous steps) DOS boots from the hard disk. From here you can switch operating systems from the DOS partitions. You do not need a bootable DOS floppy disk as long as DOS is loaded on the DOS partition of the hard disk.

To switch back to the UNIX system partition, **run fdisk** under DOS and make the UNIX partition active. To reboot the UNIX partition, press ⟨Ctrl⟩⟨Alt⟩⟨Del⟩, or the reset key, or turn the computer off, then on.

Because the UNIX system partition must be active for it to operate, you cannot use a bootable floppy to boot the operating system. This second method is appropriate for an occasional change of the active operating system.

Table 8-1 DOS hard disk devices

XENIX device convention	UNIX device convention
/dev/hd0d	/dev/dsk/0sd (linked with 0sC)
/dev/rhd0d	/dev/rdsk/0sd (linked with 0sC)
/dev/hd1d	/dev/dsk/1sd
/dev/rhd1d	/dev/rdsk/1sd

The hard disk device names in Table 8-1 are similar to */dev/hd0a* (the active disk partition) in that the disk driver determines which partition is the DOS partition and uses that as *hd0d* and *hd1d*. (You can use the XENIX or UNIX system device name conventions; they are equivalent.) This means that software that is running from the UNIX system partition and using the DOS partition does not need to know which partition is DOS (the disk driver determines that).

Installing a UNIX system partition on a DOS system

If you wish to set up your UNIX system on a hard disk which previously contained only DOS, follow these steps:

1. Copy (back up) all the DOS files and directories on the hard disk onto floppies, or whatever backup media you wish to use.
2. Run **fdisk**, under DOS. If there is enough free space for a UNIX system partition on your hard disk (check your *Installation Guide*, skip to Step 4. Otherwise, delete the DOS partition, then recreate it, leaving enough room on the disk for your UNIX system distribution and any other software that you intend to install.
3. Return the DOS files from the backup media to the newly created DOS partition on the hard disk. Keep the backups in case there is an error of some kind, so you will not lose any data.
4. Turn off your computer.
5. Follow the installation procedure outlined in the *Installation Guide* to install your UNIX system distribution.

   You will see a message warning that the contents of the hard disk will be destroyed. There is no cause for concern, because you have already backed up the DOS files and transferred them to the new DOS partition. The new partition being created will contain your UNIX system, and the installation process will only write information on the UNIX system partition.
6. During the installation procedure, **fdisk** is invoked to partition the hard disk. Use **fdisk** to assign a sufficiently large UNIX system partition.
7. Designate "UNIX" as the active operating system by choosing the "Activate Partition" option under **fdisk**.
8. Finish installing the UNIX system distribution.

NOTE UNIX **fdisk** displays DOS partitions as "DOS" while DOS **fdisk** displays UNIX system partitions as "Other". You can only create DOS partitions using DOS **fdisk**, and UNIX system partitions using UNIX system **fdisk**.
Be aware that DOS **fdisk** reports sizes in terms of cylinders, while UNIX **fdisk** reports sizes in terms of tracks. Check your hard disk manual for the number and size of cylinders on your hard disk.

Using a UNIX system and DOS with two hard disks

Your computer always boots the operating system in the active partition on the first hard disk. The UNIX system must boot from the first hard disk. There are several ways to configure your system if you have two hard disks and want to boot DOS. Two ways are discussed here.

One configuration consists of designating the entire first disk as a UNIX system partition. You then use a DOS boot floppy to start DOS and specify:

```
A> D:
```

to switch to the DOS area on the second hard disk, where **D:** is the designation for the second hard disk. This strategy works for some versions of DOS. Early versions recognize only the first hard disk on the system.

NOTE If you devote a hard disk for use with DOS, the disk must already be configured under DOS. See the *Hardware Configuration Guide* for details regarding hard disk configuration.

Another method is to maintain a small DOS partition on the first hard disk. The DOS partition is designated the active partition. In this configuration, the computer always boots DOS. This requires changing the active partition to boot the UNIX system from the hard disk.

If you use the entire second disk for DOS, you need only run **mkdev hd** to create device files for the second disk if you plan to use the UNIX system DOS utilities (**doscp**, **dosls**, **doscat**, and so on). If you do not wish to use those utilities to access DOS files on the second hard disk, there is no need to run **mkdev hd**.

NOTE Be sure to make a backup copy of your boot floppies if you use them to boot your secondary operating system.

Removing an operating system from the hard disk

You may find that you no longer need one of the operating systems installed on your hard disk. If you want to delete an operating system, use the appropriate version of **fdisk**. To delete a UNIX system partition, you must use the UNIX system version of **fdisk**. To delete a DOS partition, use **fdisk** under DOS. To delete an Extended DOS partition, you must delete all logical drives on that partition using **fdisk**. Deleting the partition removes the contents of that partition and leaves unallocated space.

You can then reallocate that space by either adding another UNIX system or DOS partition, or enlarging an existing partition. Enlarging a partition requires reinstalling the operating system and (for a UNIX system partition) remaking the filesystem on the partition using **divvy**(ADM).

DOS accessing utilities

The DOS accessing utilities are discussed in detail in "Using DOS accessing utilities" in the *User's Guide*. Note that you must have a bootable, although not active, DOS partition on the hard disk or a DOS floppy in order to use these UNIX system commands. For example, you can only transfer a file from a UNIX system partition on hard disk to a DOS floppy if either the DOS floppy is bootable or there is also a DOS partition on the hard disk.

You may also be able to use the UNIX system **dd**(C) and **diskcp**(C) commands to copy and compare DOS floppies. The UNIX system **dtype**(C) command tells you what type of floppies you have (various DOS and UNIX system types).

Also, the file */etc/default/msdos* describes which DOS filesystems (for example, A:, B:, C:, . . .) correspond to which UNIX system devices.

NOTE You cannot execute (run) DOS programs or applications from your UNIX system.

The UNIX system does not record bad tracks in the DOS area of the hard disk. If a bad track develops in the DOS area, an operation such as **doscp** that attempts to access the affected area may fail. If such is the case, the message "Error on fixed disk" is displayed.

NOTE When trying to use the DOS utilities to access files on your DOS partition, you may see the error message "bad media byte". This message indicates that the DOS partition on the hard disk is not bootable. You can make your DOS partition bootable by first backing up the files on the DOS partition, booting DOS from the floppy, and formatting the DOS partition using the command:

format c: /s

You should now reinstall your DOS files.

File and directory arguments

The file and directory arguments for DOS files take the form:

device:name

where ***device*** is a UNIX system pathname for the special device file containing the DOS diskette or DOS partition, and ***name*** is a pathname to a DOS file or directory. For example,

/dev/fd0:/john/memos

indicates that the file *memos* is in the directory */john*, and that both are in the device file */dev/fd0* (the UNIX system special device file for the primary floppy drive). Arguments without ***device:*** are assumed to be UNIX system files.

User configurable default file

For convenience, the user configurable default file */etc/default/msdos* can define DOS drive names that you can use in place of UNIX system special device file pathnames. For example, you can include the following entries in the above file:

```
A=/dev/fd096ds15
B=/dev/fd048ds9
C=/dev/dsk/0sC
D=/dev/dsk/0sD
```

Once you have defined the variables, you can use the drive letter **A:** in place of the special device file */dev/fd0* (96ds15 by default) when referencing DOS files or directories. For example:

/dev/fd0:/john/memos

can be replaced with:

A:/john/memos

The drive letter **B:** refers to a low density (48ds9) primary floppy drive. Drive letter **C:** refers to the primary DOS partition on the primary hard drive. **D:** refers to a logical drive in the extended DOS partition.

NOTE If you get the message "cannot open /dev/dsk/0sC", or a similar message, check the user permissions on the special device file involved. As super user, change the permissions with the **chmod** command. For example:

chmod 666 /dev/dsk/0sC

gives full read and write permissions to all users for the special device file */dev/dsk/0sC*, which is the DOS partition on the primary hard disk.

Mounting DOS filesystems on a UNIX system

In addition to the DOS utilities provided with the Operating System to manipulate DOS files, (described in the *User's Guide*) it is also possible to mount a DOS filesystem and access its files freely while still operating from your UNIX system.

This means that DOS files can be edited or examined in place, without first copying them into the UNIX filesystem. The major restriction is that DOS files and applications cannot be executed under this arrangement; this requires use of VP/ix (if running under your UNIX system) or booting of the DOS partition. However, data files and text files can be examined, copied or edited.

NOTE On a mounted DOS filesystem you cannot use the DOS utilities. If used, they return an error message:

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

You cannot create DOS filesystems using the **mkfs**(ADM) command. The DOS mounting feature is intended for existing DOS filesystems (as in floppy disks and an existing DOS partition).

Configuring support for mounted DOS filesystems

In order to mount DOS filesystems, the support for these features must be present in the kernel. If it is not, you must first add this to your kernel with the **mkdev**(ADM) command. Make certain you are logged in as *root* and enter the following command:

mkdev dos

Δ **sysadmsh** users select: System ➪ Configure ➪ Kernel ➪ DOS

This command adds the necessary functionality and prompts to relink the kernel. (If the link kit is not installed, you will be asked to install it.) After rebooting, you can mount DOS filesystems as described in the sections that follow.

How DOS filesystems are accessed

The Operating System deals with DOS filesystems by superimposing certain qualities of UNIX system filesystems over the DOS filesystem without changing the actual files. UNIX system filesystems are highly structured and operate in a multiuser environment. Thus they include many distinctions that have no meaning under DOS, including:

- file ownership
- access permissions
- special files (pipes, device files, etc.)
- links

NOTE Other applications and operating systems permit the mounting and access of DOS filesystems in this manner. However, most modify the DOS filesystem in some way to accomplish this. To ensure portability, there are no proprietary modifications or extensions to the DOS filesystem in this implementation. The ability to mount these filesystems is achieved purely through the facilities of the filesystem switch (FSS).

In order to make DOS files readily accessible, access permissions and file ownership are superimposed on the DOS filesystem when mounted.

Using the mount command

NOTE Make sure that you have run **mkdev dos** before attempting to use the **mount** command.

The form for a DOS filesystem mount command is:

mount -r -f DOS /dev/dsk/*xsy* /*mountpoint*

where:

x	is the hard disk number
y	is the drive letter (C:, D:, etc.)
mountpoint	is the name of the directory in the root filesystem where the DOS filesystem is to be mounted

The **-r** flag mounts the filesystem read-only, an optional precaution that will prevent damage to the DOS filesystem, which is not as robust as a UNIX system filesystem.

NOTE DOS automatically calls the primary DOS drive, on the first disk, C:. If you have a primary DOS partition on the second disk this becomes D:, automatically, and logical drives on extended partitions are named in order, for example: disk0 Primary C: EXT E: F: G: H:, disk1 Primary D: EXT I: J:, etc. The naming convention in UNIX System V, for the above example, is as follows:

DOS	C:	D:	E:	F:	G:	H:	I:	J:
UNIX	0sC	1sC	0sD	0sE	0sF	0sG	1sD	1sE

When using **mount**, you must give the specific hard disk and partition numbers (as opposed to using wildcards).

Mounting a floppy disk

You can also mount DOS floppy disks, as in the following example using the 96-tpi floppy mounted on */mnt*:

```
mount -r -f DOS /dev/fd096 /mnt
```

Repairing and checking DOS filesystems

The operating system includes a DOS version of the **fsck**(ADM) utility that works on DOS filesystems. This utility reconciles the DOS FAT (File Allocation Table) to the files contained on the filesystem. When **fsck** is invoked, it automatically detects the DOS filesystem and invokes the proper binary.

Who can access the mounted DOS filesystem

Only *root* can mount a filesystem. Access by users is governed by the permissions and ownership that *root* places on the DOS filesystem. Because of the limitations discussed earlier, DOS does not recognize permissions or ownership. When mounted on a UNIX system, the DOS files behave as follows:

- The permissions and ownership of the filesystem are governed by the mountpoint. For example, if *root* creates a mountpoint */x* with permissions of **777**, all users can read or write the contents of the filesystem. If the mountpoint is owned by *root*, all files within the DOS filesystem and any created by other users are all owned by *root*.
- The permissions for regular files will be either **0777** for readable/writable files or **0555** for read-only files. This preserves the consistency of the DOS filesystem. If a user can access the filesystem, the user will be limited by the permissions available under the DOS directory structure. This permission is read-only or read-write. When a file is created, the permissions are based on the **umask**(C) of the creator. For example, assume the user's **umask** is **022**, which generates files with permissions of **777**. Here are further examples.

Example 1: Creating a file.
The permissions are based on the **umask** owner section. A **umask** of **022** will provide a file of **777** on the DOS partition. This is because the owner has not masked off the write bit for themselves.

Example 2: Examining a file already on the DOS partition.
The permission you see is the logical AND of the UNIX system mountpoint permission and the DOS file permission. So, a UNIX system mountpoint of **750** and a DOS file permission of **555** will give you **550** for the permissions. This has nothing to do with the **umask**.

- There can only be one link for each file under the DOS filesystem. "." and ".." are a special case under this arrangement and are not links as they are on a UNIX system.
- On UNIX systems, features such as locking govern how, under certain programs and applications, a file is accessed simultaneously by different users. These features operate identically on a mounted DOS filesystem. Two users can edit the same file and write to it as permitted by the locking mechanism used.

Appearance of DOS files

As no attempt is made to change the nature of DOS files, the carriage return character (^M) will be visible when editing a DOS file on a UNIX system. (UNIX systems use only a newline, while DOS uses a carriage return and a newline.) The **dtox**(C) and **xtod**(C) commands are the easiest way to switch the end-of-line format. **dtox** is used to change DOS format to UNIX system format, and **xtod** vice-versa. These tools are described in more detail in "Using DOS accessing utilities" in the *User's Guide*.

Restrictions

Additional logical restrictions that must be observed relating to filenames, modification times, and backup utilities are as follows.

File names

The rules for file names and their conversion follows the guidelines found in the **dos**(C) manual page. In addition, the standard DOS restrictions on illegal characters apply. However, wildcards can be used just as they can with a UNIX system.

Modification times

When accessed from the UNIX system partition, the creation, modification, and access times of DOS files are always identical and use GMT, or Greenwich Mean Time. (This is because UNIX System V uses GMT internally and converts it for the user.) This means that files created in the DOS filesystem while under a DOS or UNIX system will not have consistent times across the operating systems.

UNIX backup utilities

The **backup**(ADM) utility cannot be used to make backups of a mounted DOS filesystem. DOS utilities and other copy programs like **tar**(C) will work as expected.

For more information, including more technical aspects of DOS usage, refer to **dos**(C).

UNIX systems and DOS on nonstandard disks

The UNIX system provides support for "nonstandard" hard disks. The term "nonstandard" refers to hard disks for which there are no correct disk parameter entries in your computer's ROM.

The correct parameters you specify for your nonstandard disk(s) are stored in the masterboot block, which is the first sector of your boot hard disk drive. The hard disk characteristics are specified during UNIX system installation and these characteristics are then written out with the rest of the masterboot block. The special masterboot block that comes with your UNIX system distribution resets the disk parameters to the specified values no matter which operating system is "Active." This mechanism provides nonstandard disk support for both UNIX and DOS systems.

Although the special masterboot supports nonstandard disks under DOS, you cannot use your UNIX system to install DOS on your hard disk. If a nonstandard disk is being used, it is assumed that you already have some method to transfer your DOS files to the hard disk.

Unless you are changing the active partition, you should only use the UNIX system **fdisk** to manipulate your hard disk partition table. Using DOS **fdisk** or custom **fdisk** provided by hard disk manufacturers after the UNIX system has been installed may disable non-standard disk characteristics, rendering your disk inaccessible.

Chapter 9

Maintaining system security

Every computer system needs protection from unauthorized people accessing the computer, disks, and system files. The security features present on your system represent enhancements to the basic security features of UNIX operating systems. The operating system is designed to meet the requirements of the C2 class of trust as defined by the Department of Defense's *Trusted Computer System Evaluation Criteria* (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 the "Using a secure system" chapter of the *User's Guide.*

This chapter includes information on the following:

- an overview of system security
- running a trusted system
- protecting the data on your system
- terminal login management
- activity report generation
- detecting system tampering
- dealing with filesystem corruption
- daemon operations on a trusted system
- disabling C2 features

WARNING The security features of the operating system are useless if your hardware and media are not protected. You must protect the computer itself, the distribution diskettes, and any backup media from unauthorized access. This is accomplished by the following rules:

- 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.

What is a trusted system?

Because there is no such thing as a computer system that is completely free from risk, systems are referred to as "trusted" rather than "secure". 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 your trusted system 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. Any lapses from a trusted state invite system penetrations. To be effective in your administrative position, you must understand the system's security policy, how it is controlled by system information (databases), and how changes you make affect user and administrator actions.

Trusted system concepts

The following section 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, the latter of 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. The **sysadmsh**(ADM) provides a menu-driven, administrative interface to help you maintain the TCB.

Accountability

An action is "accountable" if it can be traced to an individual person. 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). A trusted UNIX system extends the standard discretionary access control rules used by the UNIX file permissions by restricting the following:

- 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

Authorizations

An authorization is a user attribute that is 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. The TCB defines two types of authorizations: kernel and subsystem. Kernel authorizations are associated with processes. They allow a process to perform certain actions if the process has the requisite privilege. Subsystem authorizations are associated with users. They allow the user to perform a special action using a subsystem's commands (trusted utilities and programs). A "subsystem" is a related collection of files, devices, and commands that serve a particular function. For example, the *lp* subsystem consists of the print spooler files, the printer devices, and commands such as **lpadmin**(ADM) that help maintain the subsystem.

Kernel authorizations are stored in an "authorization set" associated with every process. The authorization set is a list of privileges that allow a type of action if the authorization is present, and do not allow the action if the authorization is absent. Authorizations are set either by the system defaults, or are 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.

A trusted system extends 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 (subsystem authorization *auth*), maintains this aspect of the system. This administrator's responsibilities are described in detail in later sections.

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. The trusted operating system provides an extensive series of records, or "trail," of actions. In this trail is a record of every access between subject and object (successful and unsuccessful) and every change of subject, object, and system characteristics. The audit subsystem is controlled by a separate role called audit administrator (subsystem authorization *audit*). The audit administrator decides how much information is recorded, 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 set user ID (SUID) and set group ID (SGID) mechanisms. (The ability to set user or group ID on execution is accomplished via the **setuid** and **setgid** system calls and via the **setuid** and **setgid** permission bits on files. For more information, see the **chown**(C), **setuid**(S), and **setgid**(S) manual pages.) With these you can construct programs maintaining private information. This information can only be accessed or modified by the operations implemented in the programs. The 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.)

Running 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 a single person to administer the system or a group, with each person assigned an individual subsystem
- assigning kernel authorizations to users requiring additional privileges
- deciding how strictly you wish to control and monitor access to your system
- automatically logging out idle users
- deciding how to use the auditing features

Assigning administrative roles

The first basic choice you must make is who will maintain the trusted system. You can have a single, all-powerful super user 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. The administrative tasks for a trusted UNIX system are split into a number of logical roles. Each role is responsible for maintaining one aspect of the system. The idea of specific administrative roles (and their associated tasks and responsibilities) is pivotal to your understanding of a trusted operating system. Each logical role can be assigned to the same person or to separate members of an administrative staff. Each extended role has a special authorization and a **sysadmsh** selection. That association, together with a sophisticated tracking system, enables the administrator to maintain a clear record of administrative actions. This helps to prevent problems and makes existing problems easier to identify and solve.

To perform the tasks associated with an administrative role, an administrator must have the appropriate subsystem authorization. Table 9-1 lists the subsystems, associated authorizations, and the areas of the system maintained by each role.

Table 9-1 Protected subsystems and administrative roles

Role	Subsystem authorization	Area
System Administrator	su	**su** access to other accounts
	sysadmin	*not implemented*
Audit Administrator	audit	audit databases and audit trail
Accounts Administrator	auth	system accounts
Operator	backup	filesystem backups
Cron Administrator	cron	*at* and *cron* subsystem
Printer Administrator	lp	line printer subsystem
Password Administrator	passwd	passwords
*	mem	access to process table data
*	terminal	terminal device permissions

* These are not administrative roles, but are listed for completeness.

It is vital that you understand the responsibilities for each role and the impact of your actions on the security of the system. You should configure and run the system based on the sensitivity of information kept on your site, the perceived degree of cooperation and expertise of your users, and the threat of penetration or misuse from insiders and outsiders. Only your vigilance and proper use of the system can keep the system trusted and protect the integrity of your system.

To assign a subsystem authorization, make the following **sysadmsh** selection:

Accounts ➪ User ➪ Examine:Privileges

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.

Administering subsystems with sysadmsh

Certain subsystems are logical divisions rather than actual areas of system administration. For example, the *mem* authorization is not associated with an administrative role, but it controls access to kernel memory structures. Other subsystems require administration and have **sysadmsh**(ADM) selections. These subsystems can be assigned to individuals, and documentation is provided for each area. Table 9-2 lists each of the subsystems that must be administered, their **sysadmsh** selections and the chapters that deal with them.

Table 9-2 Subsystems and sysadmsh selections

Protected subsystem	Purpose and role	sysadmsh selection	Chapter or appendix of this guide
Line Printer	Printer administration	Printers	Using printers
Backup	Filesystem copies	Backups	Backing up filesystems
Authentication	Account maintenance	Accounts	Administering user accounts
Cron	Task scheduling	Jobs	Authorizing the use of job scheduling commands
Audit	Auditing	System ⇨ Audit	Using the audit subsystem

The subsystems are described in detail in **subsystem**(M). This manual page lists all the programs and data files associated with a subsystem. Most of the functions normally exercised by the super user 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 super user. This includes mounting and unmounting filesystems, and traversing the entire file tree. Only the super user can do everything. Restrict the *root* password to a few users and assign a responsible user to the *root* account. (See the "Administering user accounts" chapter of this guide.)

Assigning kernel authorizations

As discussed previously, the TCB has two types of authorizations: kernel and subsystem. Table 9-3 contains a list of kernel authorizations.

Table 9-3 Kernel authorizations

Authorization	Action
configaudit	ability to modify audit parameters
writeaudit	ability to write audit records to the audit trail
execsuid	ability to run SUID programs
chmodsugid	ability to set the SUID and SGID bit on files
chown	ability to change the owner of an object
suspendaudit	suspends the audit of a process

Most users require only **execsuid** kernel authorizations to perform routine tasks. If the user needs to create files with the SUID or SGID bits, they must have the **chmodsugid** authorization. To change ownership of a file (give it away), the **chown** authorization is required. If a user does not have this authorization, ownership of files can only be changed by *root*. The audit

kernel authorizations (**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.

NOTE Restricted **chown** is required for NIST FIPS 151-1 conformance. The **chown** authorization should not be assigned to users if you wish to conform to NIST FIPS 151-1 requirements.

To assign a kernel authorization, make the following **sysadmsh** selection:

Accounts ⇨ User ⇨ Examine:Privileges

Users assigned administrative roles must also have certain kernel authorizations to perform the tasks required by the subsystem. The requisite kernel authorizations are shown in Table 9-4.

Table 9-4 Subsystem kernel authorization requirements

Subsystem	Required kernel authorization
audit	configaudit, writeaudit, execsuid
auth	chown, execsuid
backup	execsuid
lp	chown
cron	execsuid, chown, chmodsugid
sysadmin	execsuid, chown, chmodsugid

Controlling system access

One important aspect of operation on a trusted system is locating potential problems relating to security. The restriction mechanisms fall into three categories, all of which can be customized and reported on:

- password restrictions
- terminal use restrictions
- login restrictions

Password restrictions

The Department of Defense's *Password Management Guideline* (also known as the *Green Book*) was used as a model for password restrictions, and users are subject to much stricter password checking than traditional UNIX systems. The authentication administrator can either allow users to 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 authentication administrator.

The lifetime of a password is defined as follows:

- *The password is valid.*
- *The password has expired.* The user can still log in and change the password (if authorized to do so).
- *The password is dead.* The user is locked out and the administrator must unlock the account.

If users are not allowed to change passwords, a new password must be assigned. Users must notify the administrator when their account is locked; there is no notification mechanism other than use of regular reports on impending expirations (see "Activity report generation" (page 220) for more information). You can also use the **prwarn**(C) utility.

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 on a system-wide (System Defaults database) and per-user (Protected Password database) basis. Refer to "Changing default password restrictions" (page 87) and "Changing a user password or password parameters" (page 71) for more information.

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, which is 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. (Similar restrictions can be defined for accounts rather than terminals.) To change or examine terminal restrictions, refer to "Terminal login management" (page 217) for more information.

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 "Changing default login restrictions" (page 85) for more information.

Status reporting on access restrictions

Each of the restrictions discussed in this section have reporting facilities. For example, you can generate a report on the login records for a terminal or group of terminals, or report on user accounts with passwords that are about to expire. Refer to "Activity report generation" (page 220) for information on procedures for running these reports.

Logging out idle users

Finding a user logged into the system who has not entered any command or information for a long time can indicate that the user left the terminal and forgot 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 super user to run **idleout**.

To begin monitoring line activity for the system, enter:

idleout

Δ **sysadmsh** users select: System ➪ Configure ➪ Autologout

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, enter the command name, **idleout**, on a line by itself in the file */etc/rc2.d/S88USRDEFINE*.

Using auditing on your system

Auditing keeps detailed records of system usage, enabling you to determine if any tampering has occurred (whether attempted or successful). However, auditing can require additional supervision and disk space, depending on how long it is enabled. Auditing is discussed extensively in the "Using the audit subsystem" chapter in this guide.

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 debugging tool for programs.

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, you should refer to the *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.

Your system also includes important filesystem features that extend the protection of standard 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, discussed below, include the special use of the sticky bit on directories, data encryption, and precautions to follow when importing data files from another system.

SUID/SGID and sticky bit clearing on writes

SUID and SGID permission bits on files change the user and group IDs of a process on execution of a program. Ordinary users should not be able to set these bits, and their use is restricted by the **chmodsugid** kernel authorization. (Refer to "Assigning kernel authorizations" (page 208) for more information.) Trusted UNIX guarantees that the SUID and SGID bits are cleared on files that are written. The reason for the clearing is to prevent a user from substituting another program to take advantage of its SUID or SGID bits, which they could not otherwise set.

NOTE The clearing of SUID/SGID bits can be disabled if desired. Refer to "Disabling C2 features" (page 235) for more information.

An SUID bit shows as an "s" in the permissions of a file. In Example 9-1, the bit clearing is demonstrated twice (user input is in boldface).

Example 9-1 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 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, while both the SGID and sticky bits have a meaning for directories that warrant their remaining there. This is described next.

The sticky bit and directories

Another important enhancement involves the use of the sticky bit on directories. A directory with the sticky bit set means that only the file owner and the super user may remove files from that directory. Other users are denied the right to remove files irrespective of the directory permissions. Unlike with files, the sticky bit on directories remains there until the directory owner or super user 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 area of less secure systems. Together, both features

prevent other users from altering or replacing any file you have in a public directory. The only information they can gain from the file is its name and attributes.

Example 9-2 illustrates the power of such a scheme. The sticky bit is the "t" in the permissions for the directory. (On UNIX systems, the present directory is shown in a file listing as a dot (.), and two dots (..) represent the directory level above the present one.)

Example 9-2 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
```

(Continued on next page)

(Continued)

```
$ 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, the following:

- */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***

Using 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 "Using a secure system" in the *User's Guide*.

NOTE The data encryption software is not included in your distribution, but is available by request only within the United States. You can request this software from your dealer or distributor.

Importing 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.

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 the policies on differing systems dictate setting different modes. These considerations are critical when the imported files are system files.

To minimize your intervention and clean up after importing files, train everyone on the system to use archive program options that do not reset ownerships. The files are owned by the user importing the files. The **cpio**(C) program only preserves the ownerships of files loaded from an archive when invoked by the super user. 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, or sticky bits set. All of these situations can create security problems for you.

To minimize the effects of the archive permissions, use archive options that examine the contents without extracting anything. For example, the **tv** option to **tar** and the **-tv** option to **cpio** let you see the modes of the files on tape and prepare for any ill effects when extracting files. When bringing in unfamiliar archives, first import files into a hierarchy not accessible to others. Then manually move the files, after adjusting the ownership and modes according to your system policy.

Filesystems

Mounting filesystems that were created or handled elsewhere has all the same concerns as importing files. Filesystems also bring with them two extra concerns. The first is that the filesystem may be corrupted. The second is that file permissions on the filesystem may not be acceptable for your system. In either case, mounting a bad filesystem can cause the system to crash, the data on the imported file system to be further corrupted, or for other filesystems to go bad from side-effects. This is why the **mount**(ADM) command is reserved for the super user. The **fsck**(ADM) program should be run on all filesystems before they are mounted. If the filesystem contains system files, the **integrity**(ADM) and **fixmog**(ADM) utilities should also be run after it is mounted.

Imported filesystems can contain file permissions not suitable for your system. The super user of the imported filesystem may have set ownerships, sticky bits, special files, SUID/SGID bits, and file tree compositions incompatible with your system policies. Special files may exist with different ownerships and modes that you cannot allow.

You can 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 login management

The Terminal Control database stores parameters about system terminals. This database gives the administrator control over how many unsuccessful login attempts can be made before the terminal locks. It also stores the login activity for the terminal. When you install a terminal or printer, the information is automatically added to the Terminal Control database. However, you must modify these entries to govern how they can be used and what security procedures will be observed.

The Terminals selection of the **sysadmsh** "Accounts" branch has the following selections:

Examine	views or modifies an existing terminal entry
Create	makes a new terminal entry
Delete	deletes an existing terminal entry
Lock	locks a specific terminal
Unlock	unlocks a specific terminal
Assign	manages device name equivalencies database

Basic entries in the Terminal Control database are automatically created as tty devices are added to the system. The selections that you use most of the time are Examine, Lock, and Unlock. By default, the system manages entries as required.

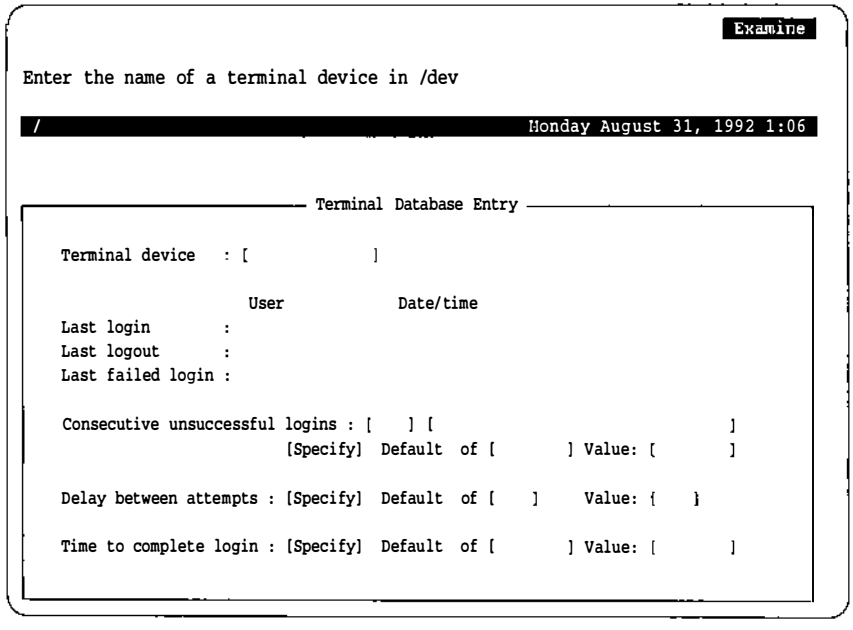
The remaining selections, Create, Delete, and Assign are special cases that are used when special hardware or software has been added to the system that requires manual configuration.

Examining a terminal entry

To modify settings for a terminal, make the following selection from **sysadmsh**:

Accounts ➪ Terminal ➪ Examine

The following screen is displayed:



This screen lets you examine the current status of the terminal. All choices require a terminal name, which is the directory entry for the terminal in the */dev* directory. A value of "INFINITE" for the "Consecutive unsuccessful logins" disables this type of lock for the terminal (abbreviations are acceptable). The "Terminal device" entry is related to the device assignment database, as described later in this chapter.

NOTE The super user can break the terminal lock on the system console. 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.

Redefining login attempt limit

If the login restriction on a terminal proves problematic, or has proven too loose, use the following **sysadmsh** selections to define these limits:

Accounts ➪ Terminal ➪ Examine

See the previous section for the form displayed. Change the "Consecutive unsuccessful logins" and "Delay between attempts" as desired.

Locking or unlocking a terminal

To lock and unlock a terminal respectively, use the following **sysadmsh** selections:

Accounts ➪ Terminals ➪ Lock

Accounts ➪ Terminals ➪ Unlock

When the prompt appears for the terminal, enter the name, for example: "tty01". When a terminal is locked, the following message is displayed when an attempt is made to log in:

```
Terminal is disabled -- see Account Administrator
```

Setting up device equivalencies database

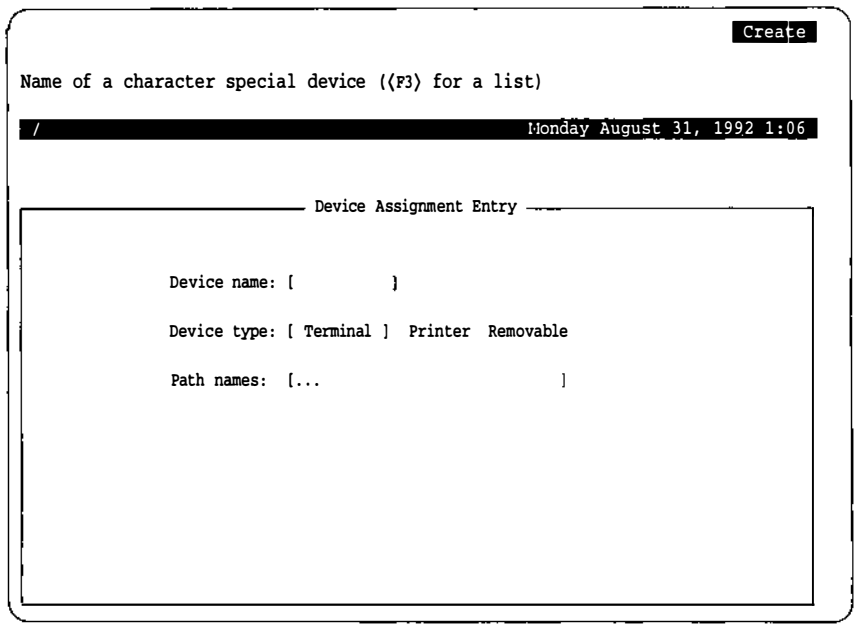
The purpose of the device assignment database is to record terminal devices that are physically the same, but referred to by different pathnames (they are linked, or are the same device with and without modem control, and so on). This equivalency mapping is very important in the case of terminals, where it ensures that the login history and terminal locking applies correctly whichever device pathname the system happens to see.

One example is someone disabling *tty1a* and then enabling *tty1A*. Because the device assignment database records the equivalence of these devices, the unsuccessful login count, for instance, is maintained. Again, the system does this automatically with devices that it recognizes by default. Any special device nodes created for unusual hardware or software have to be configured and added manually. You should only do this if the documentation has instructed you to do this or you know what you are doing.

To change a device assignment entry, select the following:

Accounts ➪ Terminals ➪ Assign ➪ Create

The following form is displayed:



Enter the device name found in */dev*. Then select the type of device, whether terminal, printer, or removable device such as a hard disk cartridge. You should then include the full pathnames of any links to the device.

Activity report generation

It is possible to create reports on the status of three important aspects of system operation:

Passwords	reports on accounts by password status
Terminal	reports on access by terminal status
Login	reports on login activity by user, group, or terminal

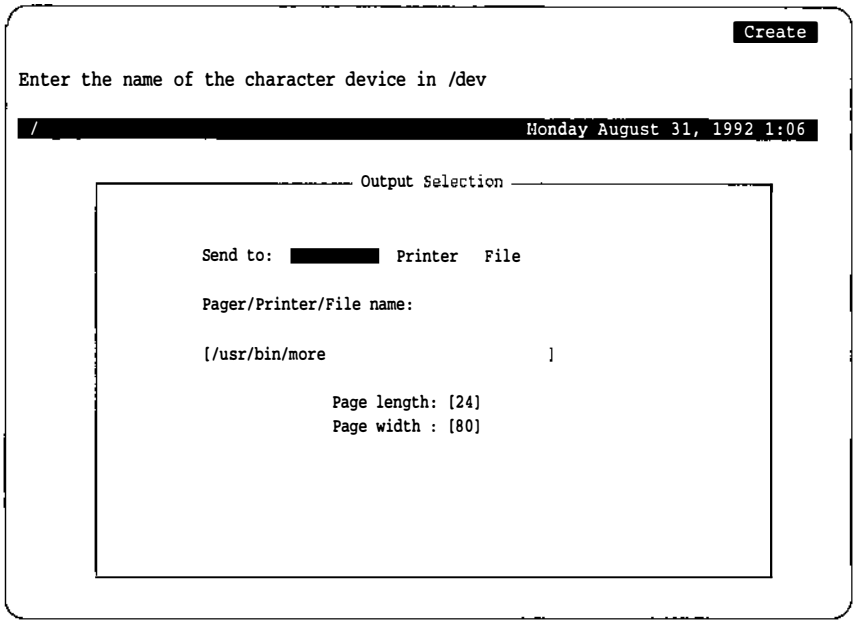
You can use the reports for security purposes (for example, listing parameters in the Protected Password and Terminal Control databases). Because these reports show system and peripheral usage, you may find them useful to fine-tune and reconfigure the system.

For all the reports, upon executing the screen you are asked to direct the output to the screen, the printer or a file.

You can filter screen output through any of the system pagination programs. The program defined by the **PAGER** environment is set up as the default; if **PAGER** is not defined, the **more**(C) program is used. For printer output, you

can name the printer device; if you do not name it, the system default printer destination is used. If redirecting output to a file, use full pathnames. No matter what category of report you select, you are always requested to define how you want the data displayed: on screen, to a printer, or into a file.

The output screen looks like this:



Reporting password status

To generate reports based on password status, make the following **sysadmsh** selection:

Accounts ➪ Report ➪ Password

Password status can be reported in several categories:

Impending	reports on accounts with passwords about to expire
Expired	reports on accounts with expired passwords
Dead	reports on accounts with dead passwords
User	reports on a single user
Group	reports on a single group of users
Full	lists all entries in password database

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.

The Impending option reports on accounts that have, or will soon have, expired passwords. This includes all accounts with already-expired passwords as well as those that will expire within one week. Although an impending expiration is not actually an error, this report lets you see users who wait until the last moment to change passwords. You may want to revise the system-wide and per-user password expiration periods based on information obtained here.

The Expired option reports on all accounts with expired passwords. These may or may not be dead passwords. All such accounts need some administrative action before the account is usable; minimally, the password must be changed.

The Dead option reports on those accounts whose password lifetime has expired, which causes the account to reject further logins.

The User option reports on the individual user that you specify. Enter the user's login name to activate it.

The Group option reports on a single specified group. This report includes all the users who belong to the specified group.

Finally, the Full option reports statistics for all users on the system.

The reports use the following abbreviations:

Dflt	Default.
Y, N, D	Yes, No, Default. Some selections have three possible values: yes, no, and the default value used by the system, which can be either yes or no.
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?

Example report: group

Example 9-3 is a sample report on the password activity of group *hamster*. The abbreviations under "Password Parameters" correspond to the system-wide default password parameters.

Example 9-3 Sample password database report by group

```
                    Password Database Report
                         System  unix
                    Wed Mar 22 10:56:29 1991

                        Password Parameters
[1] User Name Type      Min  Exp  Life Rnd? Pck? Rst? Lck?
    ---- ---- ----      ---  ---  ---- ---- ---- ---- ----
          Last Changes              Last Logins        Consec
[2] Success     Failed       Success     Failed       #Failed
    -------     ------       -------     ------       -------

[3] Kernel Authorizations
    ------ --------------
[1] alvin     general  Dflt Dflt Dflt D    D    D    Y
[2] 05/22/90    NEVER       05/22/90    NEVER       -
[3] DEFAULT
[1] simon     general  Dflt Dflt Dflt D    D    D    N
[2] 05/22/90    NEVER       05/22/90    NEVER       -
[3] DEFAULT
[1] theodore  general  Dflt Dflt Dflt D    D    D    N
[2] 05/22/90    NEVER       05/22/90    05/22/90    -
[3] DEFAULT
```

Reporting terminal activity

To generate reports based on terminal activity, make the following **sysadmsh** selection:

Accounts ➪ Report ➪ Terminal

This allows you to get statistics on the Terminal Control database. The report contains any lock conditions, unsuccessful attempts to log in at the terminal, and the delay between login attempts. Similar to the Password reporting, you can select the report to apply to a single terminal, a range of terminals, or all terminals.

When you select a user or group, the report includes both the last successful and the last unsuccessful login. The number of unsuccessful attempts is also reported. As this number 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.

When you select one or more terminals, the report includes the last successful, last unsuccessful, and last logouts on the terminal. The number of unsuccessful attempts on this terminal is also reported. Both report types can provide you with valuable data on how the system is being used.

Example 9-4 is a sample output of a terminal report.

Example 9-4 Sample terminal report

```
                Terminal Database Report
                      System  unix
                Wed Mar 22 10:58:42 1991

            Admin Login  Unsucc  Max Unsuc
Tty  Name Lck?  Delay Attempts Attempts
---  ---- ----  ----- -------- --------
console   D     Dflt  2        Dflt
tty02     D     Dflt  None     Dflt
tty03     D     Dflt  None     Dflt
tty04     D     Dflt  None     Dflt
tty05     D     Dflt  1        Dflt
tty06     D     Dflt  None     Dflt
tty07     D     Dflt  None     Dflt
tty08     D     Dflt  None     Dflt
tty09     N     Dflt  None     Dflt
tty10     D     Dflt  None     Dflt
tty11     D     Dflt  None     Dflt
tty12     D     Dflt  None     Dflt
```

Reporting login activity

To generate reports based on login activity, make the following **sysadmsh** selection:

Accounts ➪ Report ➪ Login

Login reports can be generated in three categories: by User, Group, and Terminal.

Example 9-5 is a sample output listing login attempts by terminal.

Example 9-5 Sample login report by terminal

```
                              Login Activity Report
                                  System  unix
                              Wed Mar 22 14:43:53 1991

          Last Good Login     Last Bad Login      Last Logout
Tty Name User Name Date      User Name Date      User Name Date      #Failed
--- ---- ---- ---- ----      ---- ---- ----      ---- ---- ----      -------
console  alvin     05/22/90 UNKNOWN   05/22/90 alvin     05/22/90 2
tty02    root      05/21/90 root      05/21/90 root      05/19/90 0
tty05    maryt     05/21/90 UNKNOWN   05/21/90 root      05/19/90 0
tty04    root      05/19/90 root      05/13/90 root      05/19/90 0
tty05    UNKNOWN   NEVER    root      05/13/90 UNKNOWN   NEVER    1
tty06    UNKNOWN   NEVER    UNKNOWN   NEVER    UNKNOWN   NEVER    0
tty07    UNKNOWN   NEVER    UNKNOWN   NEVER    UNKNOWN   NEVER    0
tty08    UNKNOWN   NEVER    UNKNOWN   NEVER    UNKNOWN   NEVER    0
tty09    UNKNOWN   NEVER    UNKNOWN   NEVER    UNKNOWN   NEVER    0
tty10    UNKNOWN   NEVER    UNKNOWN   NEVER    UNKNOWN   NEVER    0
tty11    UNKNOWN   NEVER    UNKNOWN   NEVER    UNKNOWN   NEVER    0
tty12    UNKNOWN   NEVER    UNKNOWN   NEVER    UNKNOWN   NEVER    0
```

Detecting system tampering

No system can be considered completely secure. When you consider that system penetration can be as simple as someone using an obvious password or leaving their terminal logged in overnight, you can see the user is the weakest link in the scheme. 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 a user has the proper authorization, then they can use system programs to modify the security databases (for example, the audit administrator changing the audit configuration, or the accounts administrator changing 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 someone gains access to an account having greater authorization than their own. After having set up your system to minimize the possibility of tampering, the remaining task is to discover whether any tampering has taken place. Tampering can come from three avenues:

- Someone obtains a password of another user or gains access to their account (as when someone leaves their terminal logged in).
- A user with authorization abuses their privileges.

- A knowledgeable user gains unsupervised access to the computer system itself.

This section discusses how to detect such occurrences.

Stolen passwords

Each time a user logs in, the system displays the time and date of their last login. 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 account usage guidelines discussed in the "Using a secure system" chapter of the *User's Guide*. These procedures ensure that other users cannot guess their passwords or otherwise obtain them.

The administrator should carefully consider what 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 a malicious 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 (refer to "Activity report generation" (page 220) for information). Look for the following:

- logins made at off hours, or at times when the user is not supposed to be onsite
- 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, which 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 access to the computer itself

The most basic security requirement is to prevent unsupervised access to the system itself. Although it is often necessary for users to access the console to operate the floppy 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 would be 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 the following:

- login reports for the *root* account
- suspicious super user 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 security perimeter. 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 itself, its users, its administrators, and its subsystems so that a site can control its own security parameters. These databases reside on the system and are maintained by an administrator. The format of these files is discussed in **authcap**(F).

WARNING The Authentication database files are not meant to be edited by hand. The trusted system utilities and **sysadmsh**(ADM) selections 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, as described later.

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 store a list of users that 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. It enhances accountability of administrative actions by allowing only specified users to run programs that maintain the internal subsystems. Security is enhanced by controlling who has permission to execute programs that maintain subsystems and by accounting for the real users that assume administrative roles.

File Control — helps maintain the integrity of the Trusted Computing Base. It does this by maintaining a record of the contents and protection attributes of files important to the TCB's operation. This database provides an effective tool for detecting modifications to the active copy of the TCB. The system administrator program **integrity**(ADM) checks the TCB file permissions against this database.

Checking the system after a crash

Several programs are used to maintain the Authentication database, the system area of the filesystem, and the filesystem as a whole. The basic rule is to work from the most basic components of the filesystem outward. Otherwise, corrections made at the higher levels may be undone by programs fixing the lower levels. Given this, use the programs in this section after a system crash or abnormality in this order:

1. Run a filesystem check with **fsck**(ADM) (occurs automatically at reboot after a crash).
2. Check for the absence of critical files with **tcbck**(ADM) (automatic at reboot time).
3. Generate an audit report (optional).
4. Check the consistency of the Authentication database with **authck**(ADM).
5. Check system file permissions with **integrity**(ADM).
6. Fix permissions with **fixmog**(ADM).

These programs should be run while the system is in single user (system maintenance) mode.

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, the following message is generated:

```
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, the following message is generated:

```
Account is disabled but console login is allowed.
```

The tty used should be physically secure; remember that normal locks do not apply to the super user account on this tty.

Filesystem checking: fsck(ADM)

Filesystems containing sensitive files must be considered sensitive entities themselves. Thus, the integrity of the filesystem afforded by the **fsck** program enhances the overall security of the system.

The **fsck** program must be run after any system crash or abnormal system termination. As always, make sure the system is in single-user mode when running **fsck**. There may have been user, system, or audit files in the process of being built when the system crashed. Although that data may be lost, **fsck** can recover some of those files in the *lost+found* directory of the filesystem, and at least fix basic filesystem problems.

Refer to "Repairing filesystems with fsck" (page 117) for more information.

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. This check is described in the "Checking the security databases" section of the "Starting and stopping the system" chapter of this guide.

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)

The **authck**(ADM) program checks the consistency of the Authentication database. (The functions of **authck** can also be accessed via **sysadmsh**, as discussed below.) There are several options that restrict the scope of the checking. See **authck**(ADM) for more information.

The Accounts menu of **sysadmsh** includes functions to check and repair inconsistencies in the Authentication database, a collection of files containing information about the trusted system.

Checking the authentication database files

To check the consistency of the password, terminal, and subsystem databases, make the following selection from **sysadmsh**:

Accounts ➪ Check ➪ Databases

This command 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. The Accounts ➪ Check ➪ Databases selection does not allow such repairs. If inconsistencies are found, you can execute **authck** from the command line or restore files if necessary.

Checking the /etc/passwd and /etc/group files

To check the consistency of the */etc/passwd* and */etc/group* files, make the following selection from **sysadmsh**:

Accounts ➪ Check ➪ Password

This selection does the same checking that is done when a new user is created. 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, however, alter permissions (refer to "System file permission repair: fixmog(ADM)" (page 232) for information on fixing permissions.)

NOTE If your system is configured with the Low or Traditional security defaults, permission problems reported by **integrity** will 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 the File Control database. You should always use the **-i** (interactive) option to ensure that you can confirm any changes before they are made.

Daemon operations on a trusted system

This section notes the features that affect system daemons, and lists examples of procedural and programming changes that must be made to add and run new daemons properly on a trusted system. This ensures 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.

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 (for example, **auditd**(ADM)) or assume that their LUID was stamped before they run (for example, **lpsched**(ADM)). 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. Refer to "Disabling C2 features" (page 235) for more information.

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, as shown below, that runs 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

Note that the standard output and error of the sample **lpsched** command was redirected to the null device. 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 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. Refer to "Disabling C2 features" (page 235) for more information.

Processes in the operating system run with a set of kernel authorizations that control the special rights the process has for certain privileged 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 "Authorizations" (page 204) for more information on kernel authorizations.) If a daemon executes other SUID programs, it must have the **execsuid** authorization. If the process creates files with the SUID bit, it must have the **chmodsugid** authorization. If it uses **chown** to alter ownership of files, it must have the **chown** authorization. No processes that are not installed with the TCB should run with any of the audit authorizations. Other authorizations are for special situations, and should not be allowed to non-TCB daemons.

Sticky directories

The final feature that may affect daemons is sticky directories. If a directory's mode includes the save text (sticky) 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 of 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.

You must carefully consider each daemon program so that it 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.

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. 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 **sysadmsh** selection System ➪ Configure ➪ Kernel ➪ Parameters and selecting category 14: "Security," and changing the parameter desired. The kernel must then be relinked and booted for the new behavior to take effect. Use the **sysadmsh** System ➪ Configure ➪ Kernel ➪ Rebuild selection to relink the kernel.

Chapter 10

Using the audit subsystem

The audit subsystem records security-related events that occur on a system in the form of an "audit trail" that can later be examined. Audit trails produced by this subsystem can detect penetration of the system and the misuse of resources. The audit subsystem is designed to meet the audit goals specified by the U.S. National Computer Security Center.

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.

The audit subsystem uses system call and utility usage to classify user actions into event types. These can be used for selective audit generation and reduction. One such event type, *Discretionary Access Control* (DAC) Denial, records attempts to use objects in a manner not permitted by the object's permissions. For example, if a user process attempts to write a read-only file, a DAC Denial event is audited, showing that the process tried to write a file to which it was not entitled. When you examine the audit trail, it is easy to notice repeated attempts to access files for which permission is not granted. This alerts the administrator to possible tampering or penetration.

Essential to the effectiveness of the audit data is the ability to uniquely identify all users and their actions so that the audit trail accurately reflects the auditable actions of each user. As users attempt to log onto the system, they must go through an identification and authentication process before access to the system is granted. The security mechanism stamps each process created by a user with an immutable indicator of identity: the login UID or LUID. The LUID is preserved regardless of transitions between user accounts with commands like **su**(C). Each audit record generated by the subsystem contains the LUID together with the process's effective and real user and group IDs. As a result, users can be held strictly accountable for their actions.

The audit subsystem is administered by the audit administrator. The audit administrator has complete control over the events selected for audit record generation, over the parameter values for subsystem control, and over the subsequent reduction and analysis of audit data.

This chapter explains the following:

- introduction: understand the principles of auditing and the design of the audit subsystem
- data collection: select audit criteria, enable and disable auditing, and adjust audit performance parameters
- file and directory management: set storage of audit records, back up and remove audit records, monitor disk usage
- report generation: use audit report templates, generate audit reports, and interpret the data

An audit glossary is included at the end of this chapter to explain the terms used.

Audit subsystem components

The Audit Subsystem consists of five major components:

- kernel audit mechanism
- audit device driver (*/dev/auditr* and */dev/auditw*)
- audit compaction daemon (**auditd**(ADM))
- **sysadmsh** audit interface
- data reduction and analysis facility

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 contains an entry in a subsystem table that indicates whether the call is security-relevant and, if so, to what event type the system call corresponds. Additionally, a table of error codes further classifies the system calls into specific security events. The kernel audit mechanism makes an internal call to the device driver to write a record to the audit trail.

For instance, the **open**(S) system call is classified as a *Make object available* event. If user *blf* performs an **open**(S) on */unix* and it succeeds, an audit record is generated indicating 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, therefore, possible that a system call might be audited selectively, depending on the event types that you enable.

Some system calls are not considered relevant to security. For instance, **getpid**(S) retrieves the process ID 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 the following:

- 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** kernel authorizations. This limits access to the audit device only to trusted utilities such as the audit daemon and the audit administrator interfaces. The audit device 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 (configurable by the administrator), the subsystem creates a new collection file and begins writing to it. The audit trail could, therefore, be viewed as a continuously growing sequential file even though many collection files are used. That is precisely how the audit trail is viewed by the audit daemon, because it reads the device and is presented with records from the audit trail. The subsystem handles the necessary switching to new collection files for the daemon when the end of a file is reached. All of this is transparent to the daemon.

Audit compaction daemon

The audit daemon **auditd**(ADM) is a trusted utility that runs as a background daemon process whenever you enable auditing. The daemon is the sole reader of the audit device which, in turn, provides the daemon with blocks of records from the audit collection files. The daemon is not concerned that the audit trail is spread over numerous collection files. The audit device driver satisfies the read requests from the daemon and handles the switching and deletion of collection files as needed.

The main purpose of the daemon is to provide a compaction and logging mechanism for the audit session. The daemon also serves a support role for protected subsystems, enabling them to write audit records to the subsystem. Depending on the audit record generation criteria you select, a large amount of audit data can be generated on the system. For a typical single-user system, it would not be uncommon to generate 200 Kbytes of audit data in an hour. The daemon, therefore, provides a compaction mechanism, compressing the audit data into a packed record format that is stored in an *audit compaction file.* The compaction algorithm provides for an average 60% reduction in file space. This 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 switches collection files as they reach specified sizes, the daemon can create multiple compaction files to avoid growing a single file too large to be manageable. (This is also configurable.) Audit compaction files written by the daemon may also be located in a variety of administrator-specified directories. For these reasons, the log file is maintained to provide a trail of compaction files that 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** authorization. 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 subsystem interface

sysadmsh presents simple options to set up and maintain the audit subsystem. This 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.

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 discarded for the purpose of that reduction.

As an example, you may reduce the data selecting the DAC Denial event with user ID *blf* looking for the object */unix*. Only records that reflect an access attempt to */unix* by *blf* that was denied because of lack of permission are printed. This provides a powerful mechanism for identifying security events of immediate interest without having to analyze the entire audit trail.

Audit methodology

This section explains how the audit subsystem functions, what criteria are used to collect data, and how audit requirements affect system performance.

Audit authorizations

There are four authorizations 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 System ⇨ Audit ⇨ Report functions.

Audit record sources

The audit trail contains the security-related events for the system. Effective auditing concerns not only system call requests from user processes but also certain events such as login, logoff, and login failure attempts. These events are critical to determining who has accessed the system, at what times, from what terminal, and what actions were performed. Login failures are impossible to audit at the kernel level 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.

Audit records are generated from three sources (discussed in the sections that follow):

- kernel audit mechanism
- trusted application processes
- authorized subsystems

Kernel audit mechanism

A large percentage of the audit records stored in the audit trail are generated by the kernel audit mechanism. This portion of the audit subsystem generates records in response to user process system calls that map to security-related events. Some system calls, **open**(S) for instance, map to multiple security events depending upon user arguments and the state of the file being opened. If **open**(S) is called with the **O_CREAT** flag, the file is created if it does not exist. If the **O_TRUNC** flag is specified, the file is truncated to zero length if it exists. This illustrates how the **open**(S) call could map to one of three distinct events, *Make Object Available*, *Object Creation*, or *Object Modification*.

Error codes also play an important role in determining the event. Errors on system calls that indicate access or permission denials as well as resource consumption problems are mapped to specific event types. The kernel audit mechanism determines at the end of the system call what event class the call belongs to, and if that event is to be audited as specified by you. 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) 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 instance, to write a login audit record to the audit trail rather than letting a login on the system be represented as a collection of system calls required to complete the login procedure.

It is not sufficient to just let the trusted applications write to the audit device. There must also be a way to suppress the generation of system call audit records by the kernel audit mechanism to avoid the problem of a cluttered audit trail. Thus, the **suspendaudit** authorization exists as discussed earlier. Trusted applications run with this authorization enabled, suspending kernel system call auditing for that process and allowing it to open and write the audit device. Only a few trusted applications are permitted to do this. A user process should not run with **suspendaudit** authorization. The authorization mechanism is managed by **login**, using restricted system calls, and is based on Protected Password database entries.

Authorized subsystems

A third method in which audit records are generated is through authorized subsystems such as the *lp*, *cron*, *terminal*, and *mem* subsystems. Sometimes, a subsystem encounters inconsistencies or problems that make the writing of an informative audit record desirable. However, subsystems do not possess the **writeaudit** authorization and cannot directly write audit records to the subsystem.

Instead, the subsystems format the records just as a trusted application would, and present the records to the audit daemon process through a 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** authorization to these systems.

Accountability for audit

The audit subsystem audits security-related system events and associates the events with a specific user. Users log into the system through the **login** program. This program performs authentication on the user to determine whether access is permitted. The login procedure has been enhanced to provide audit support for both successful and unsuccessful login attempts. When a user is successfully logged in, **login** stamps the user process with 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 as previously discussed. For application or subsystem auditing, the process writing the audit record sets the event type. This event type is not changed by the audit device or by the audit daemon.

Event types are important because they classify the security event on the system. Both audit-record generation and reduction can be controlled based on event types. For example, if you are only concerned with users logging onto and off 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 10-1; the letters are a simple identifier used to refer to the event by the audit subsystem.

Table 10-1 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*, as explained below. The audit subsystem also maintains a mask of event types for each process on the system (explained in a later section).

Mandatory auditing

To maintain accurately all the required information about a user process for meaningful audit output, the kernel audit mechanism always audits certain system calls. When auditing is enabled, this means that some events are audited even if no events were selected by the audit administrator. These are known as mandatory system calls. They are essential to the maintenance of the process state. For example, the **open**(S) system call may specify a relative pathname such as *../newfile*. The full pathname depends on the current directory of the process, which is set using the **chdir**(S) system call. The audit record containing the pathname *../newfile* could not be meaningfully reduced without prior knowledge of the value of the current directory.

The problem applies to the **close**(S) system call as well. This system call requires only a file descriptor as the argument to close a previously opened file. The **close** audit record would be insignificant unless the name of the object being closed is output in the record. However, unless the pathname is retained when the file is opened, there is no way to provide the pathname for the close. Table 10-2 lists the audit event types affected.

Table 10-2 Mandatory audit events

Event type	Always audited	Optionally audited
Make object available	open, pipe	mount, opensem
Object creation	creat	link, mkdir, mknod, creatsem, sdget
Map object to subject	dup, exec, exece	fstatfs, getdents, stat, statfs
Object modification	execseg, unexecseg	chsize, stime
Make object unavailable	close	sdfree, umount
Process create/delete	exit, fork	–
Process modification	chdir, chroot, proctl, security, setgid, setpgrp, setuid	–

Mandatory auditing is not limited to just the group of system calls listed in Table 10-2. The login event is the only mandatory trusted application audit record defined. When a user logs in, the login record contains an indicator of the terminal on which the login occurs. If that same user is logged into multiple terminals on the system, the actions of that user can be traced to a specific terminal.

System audit event mask

The system event mask is global to the audit subsystem. You 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 (using a bit-wise operation) to determine if a newly created record is enabled for auditing. The audit subsystem uses the system event mask to compute user masks when a new process is created through a login.

User-specific and process event masks

You can override the system-wide event mask for any user by setting up a *user-specific event mask*. 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 as discussed here.

The user-specific event mask has one of three values for each audit event type:

- always audit this event
- never audit this event
- use the system audit event mask

For each audit event type, the process audit mask is set from the user-specific mask if it indicates that the event is always or never audited. Otherwise, the process audit mask is set from the system audit event mask. In most cases, the user-specific event mask 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

You should follow certain guidelines to use 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

(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 provides disk-space savings benefits because the amount of audit records written to the collection files by the subsystem is reduced. There is, however, a drawback to using preselection. If a system security violation occurred and that event or the user that perpetrated the event was not selected for audit, the record of the action is lost.

For this reason, it is more conservative not to preselect the audit events and users or groups, but instead to perform full auditing. The benefit is that any security-related event that occurs is recorded in the audit trail. The disadvantage of full auditing is that it consumes a great deal of disk space.

You can then combine full auditing with *postselection* to examine only records of interest. Post-selection provides for the selective examination of the audit trail based on event types, user IDs, group IDs, and object names, as well as date and time of record generation. In all, the audit subsystem combined with the data reduction/analysis utility provides you with the flexibility to trade between system performance and disk capacity with preselection, and the convenience of full auditing combined with post-selection.

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.

How well the subsystem meets your goals depends on proper administration of the system. You control the tradeoff between reliability and performance using audit parameters. Improper setup can result in poor performance, loss of audit data, or both. For example, setting the audit event mask to govern event types audited by the subsystem is critical. For instance, if event 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 the following three items:

- performance goals
- reliability goals
- 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 where multiple clerks are accessing a common database. Lockout mechanisms must exist to prevent the intermixing of audit records and to insure the consistency of the database. The same is true of the audit subsystem collection files.

An internal buffering mechanism and a write-behind strategy tries to minimize the impact of multiple, simultaneous writers to the collection file. This lets the subsystem service audit records from processes and applications while collection files are being written in parallel. You can tune this mechanism for how much buffering is used and how frequently data is written to the collection file.

Reliability goals

Equally important to the system's performance is the reliability of the audit trail produced. Traditional UNIX systems lack the element of preserving 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 instance, the system may be limited in use to a small group of people but left unattended at night. Additionally, 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 log-in 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. However, it is often better to have recorded the events and to use the reduction tools to discard unwanted records later than not to have the records that are really needed to examine a problem. This decision depends on the degree of security you wish to 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. A session is defined as the sequence of collection and compaction files containing the audit records associated with a specific time interval. Some processes that are audited in the second or subsequent session might have been created during the first session. Consequently, a session may not contain all of the relevant process state needed for a certain process. In turn, this can lead to incomplete record reduction. This applies mostly to filenames and typically only in the case of relative (rather than absolute) filenames. You can avoid this problem by disabling auditing only by taking the system down. Refer to "Maintaining audit trail continuity" (page 256) for more information.

Administrative concerns

This section discusses particular 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, despite every effort to provide a resilient base. 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. However, this does not guarantee that data always makes it to the disk. This is especially true if a disk failure causes the system crash.

It is not uncommon to find filesystem damage on audit trail files upon 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 System ➪ Audit ➪ Files functions of **sysadmsh** before auditing is reenabled.

Auditing as a debugging tool

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.

Data collection

This section explains how to set up, activate, manage, and deactivate auditing. The system is distributed with default collection parameters (refer to "Default account configuration" (page 81) for a list of defaults). You can modify these defaults to fit your needs.

As discussed in the introduction, usage of the audit subsystem has two stages, collection and reporting. Each stage involves selection of audit data. Data collected by the audit subsystem is governed by a set of parameter files called masks. The word "mask" is used because unwanted data is "masked out", and not collected. 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.

The following data collection functions are available:

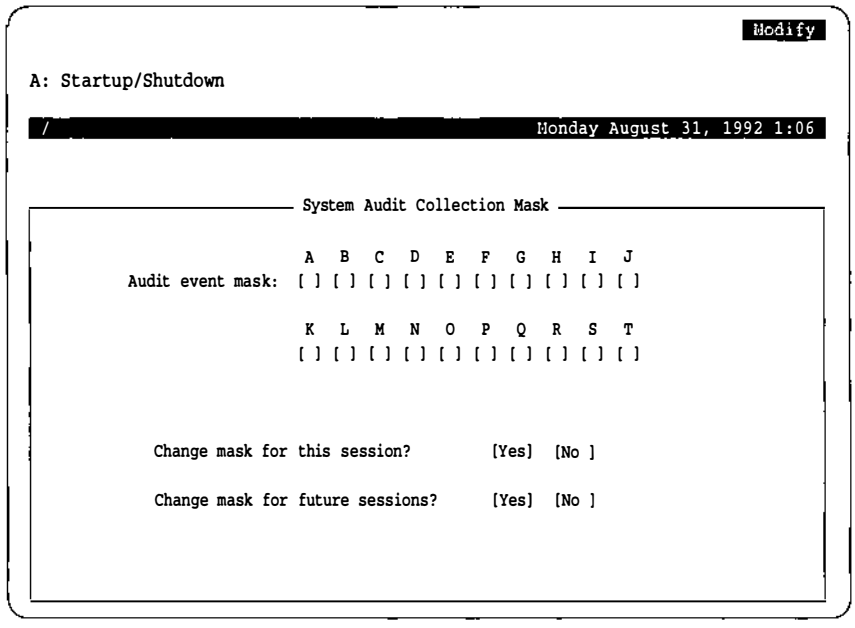
Directories	display or modify audit collection and compaction file directory list
Events	display or modify system audit collection type masks
IDs	display or modify list of users and groups audited
Parameters	audit subsystem performance parameters
Reset	change collection rules back to the default values
Statistics	display statistics of current audit session

Choosing audit events

To select events to audit on a system-wide basis, make the following **sysadmsh** selection:

System ➪ Audit ➪ Collection ➪ Events ➪ Modify

You see a screen similar to the following:



Each event type displayed corresponds to a letter. For those events that are to be audited, the event type should be specified with a "Y". Those event types that are not to be audited are excluded using the "N" option. Use ⟨Space⟩ to toggle an entry from "Y" to "N" and vice versa. Use the arrow keys to move from entry to entry. This event mask can be modified and dynamically altered for the current audit session and/or can be written to the parameter file to take effect on future audit sessions.

As discussed previously under "Audit event types," there are a number of audit events that can be selected; these are summarized in Table 10-3.

Table 10-3 Audit event types

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	super user-only actions

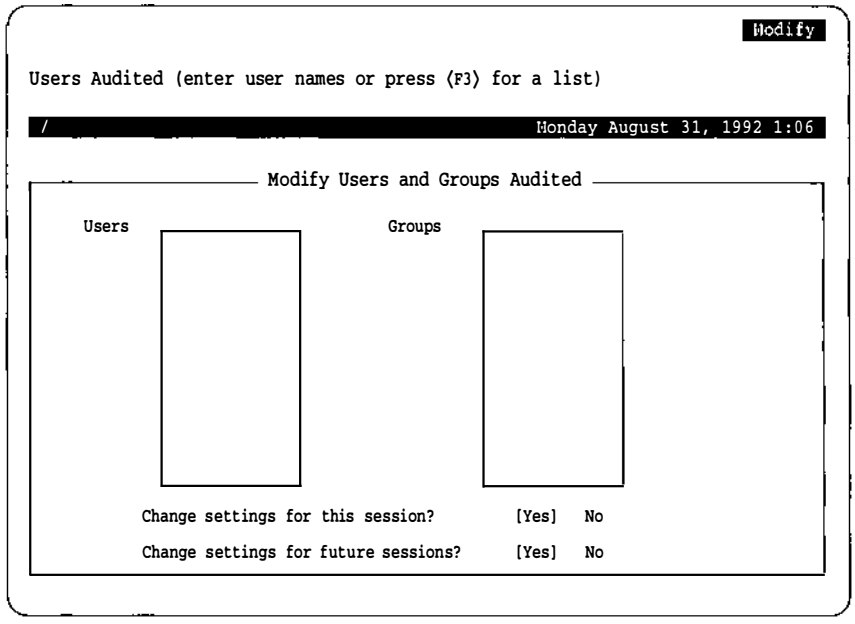
Auditing individual users and groups

The User and Group fields can dynamically alter the audit selection for the current session or can affect the next session. Selection of 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 audit by users or groups, make the following **sysadmsh** selection:

System ➪ Audit ➪ Collection ➪ IDs ➪ Modify

A screen similar to the following is displayed:



Displaying current audit statistics

An option is provided for the retrieval of the current audit session statistics, providing information on the current session number, the number of collection and compaction files, the number of records written by the kernel audit mechanism and the number written by applications, as well as other information. If auditing is not currently in effect, no statistics are displayed.

To display the statistics of the current audit session, make the following **sysadmsh** selection:

System ⇨ Audit ⇨ Collection ⇨ Statistics

Example 10-1 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

To switch auditing on or off, use the following **sysadmsh** selections:

System ⇨ Audit ⇨ Enable
System ⇨ Audit ⇨ Disable

The enable function uses the current audit parameter file to perform the subsystem initialization. The disable function is available from the same menu and causes a graceful exit from auditing (at which point all collection files have been read by the daemon and compacted). The daemon then terminates leaving only an audit session log file and the session compaction files.

Most subsystem parameters can be modified while auditing is running, so you do not need to disable audit for that purpose. Both enable and disable functions have confirmation screens that must be acknowledged before the function is completed by **sysadmsh**. When auditing is enabled or disabled, a message is displayed indicating the status of auditing at reboot time; if disabled, auditing will be disabled at system startup and if enabled, auditing will be enabled again at startup.

Maintaining audit trail continuity

There is an important consideration involving LUIDs and audit sessions. Example 10-2 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 10-2 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
```

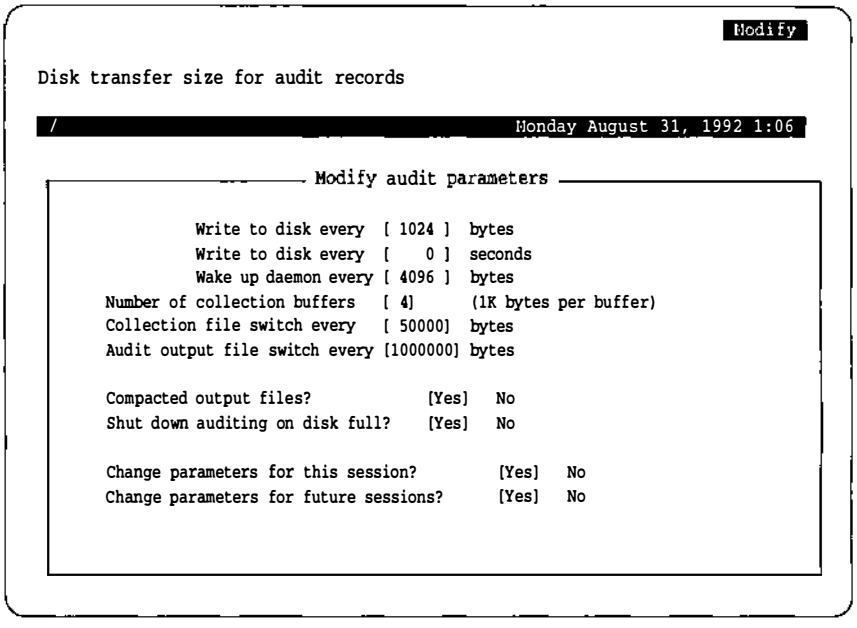
Note the (*INC*) next to the process ID. This indicates that the audit trail for this process is incomplete. It means that auditing was started *after* this user logged in, therefore there is no record of the LUID being set, and the reduction program does not know what it is. The reduction program assigns 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

You can alter some audit parameters to tailor auditing to the needs of a system. To examine the current audit parameter settings, make the following **sysadmsh** selection:

System ➪ Audit ➪ Collection ➪ Parameters ➪ Modify

A form similar to the following is displayed:



Some of these parameters relate to the earlier discussion on performance and reliability tradeoffs. This should become more apparent now. The parameters are as follows:

Write to disk 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 that can later be reduced. To maximize the effectiveness of the compaction algorithm, the daemon needs to read blocks of data between 4 and 5 Kbytes. This requires special handling by the subsystem as a typical process read returns when any data is available rather than waiting for a specified amount of data to accumulate. For maximum effectiveness, this parameter should be left at the default value of 4 Kbytes. Values greater than 4 Kbytes will not yield significant improvement.

Number of collection buffers

This lets you specify the number of collection buffers for the subsystem to use. It uses these internal collection buffers to gather audit data for writing to the collection file. Multiple buffers are used to increase the efficiency of the system 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. A minimum of two buffers is required. Most systems cannot effectively use more than 4-6 buffers to avoid performance problems. There is no deterministic way to calculate the optimum number of buffers. Generally, base this value on the expected process load of the system.

Collection file switch every [ ] bytes
Audit output file switch every [ ] bytes

These two parameters let you specify the maximum size that collection and compaction files may grow before a new file is created. Choosing a small value for either parameter results in excessive file switches. 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 50 Kbytes, and the compaction files are 1 Mbyte. 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 should non-compacted audit files be desired. There is no compelling reason why this option should be exercised because compaction does not require large amounts of additional processing time and the resultant disk savings are typically greater than 60 percent. The compaction algorithm is contained in the audit daemon user process, not performed in the kernel portion of the subsystem.

Enable audit on system startup

This option starts auditing automatically each time the system is rebooted. This field is only displayed with the View option; it is set according to whether auditing 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

The last two options on the screen let you dynamically alter the current session and/or make the changes a permanent part of the audit parameter file for future sessions.

File and directory management

This section explains how to choose the location of audit record files, back up and remove old records, and maintain the disk space used by the audit subsystem.

Audit data is divided into sessions, with a new session started each time auditing is stopped and started once again. When examining or manipulating audit data, it is done by session number.

The following session file functions are available:

List	list audit session files on the system
Backup	back up an audit file session to backup media
Delete	remove an audit session file
Restore	restore an audit file session from backup media

An audit session consists of a session log file and a group of compaction files generated between an enable and disable of the audit subsystem. Each collection file and compaction file created during a session is uniquely numbered with the session in which it was created. When sessions are completed, only the log file and the compaction files remain. The file maintenance functions examine which sessions are still on the system and let you remove sessions no longer wanted.

The audit directory maintenance functions are as follows:

List	display audit directory list
Create	create new audit directory
Delete	delete existing audit directory
Add	add entry to audit directory list
Remove	remove entry from audit directory list

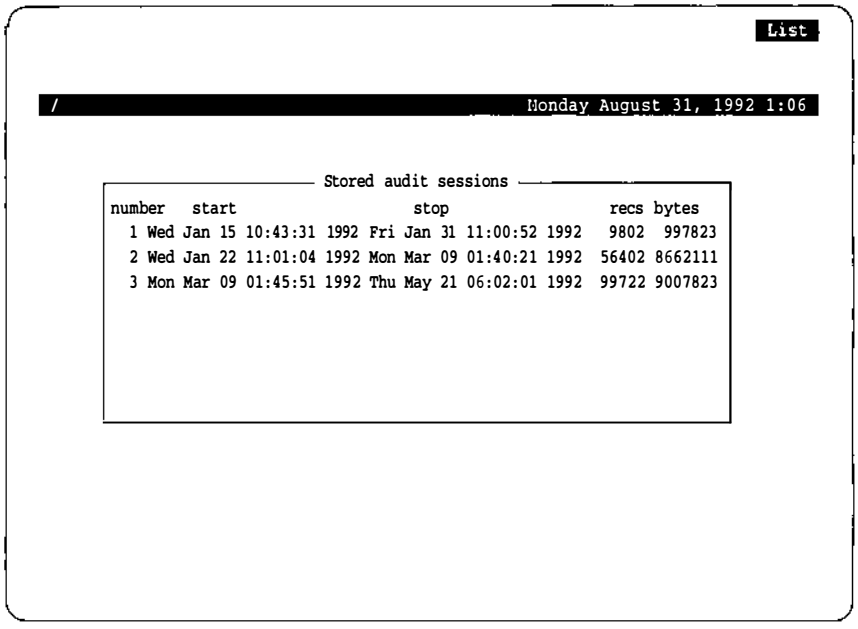
The last two functions manipulate the list of directory entries used by the audit system rather than the directories themselves.

Listing audit sessions

To list the audit sessions that are currently on the system, make the following **sysadmsh** selection:

System ➪ Audit ➪ Files ➪ List

A screen similar to the following is displayed:



The number in the first column is the session number. This list is what is displayed when you ask for a list of session numbers from other options of the audit menus.

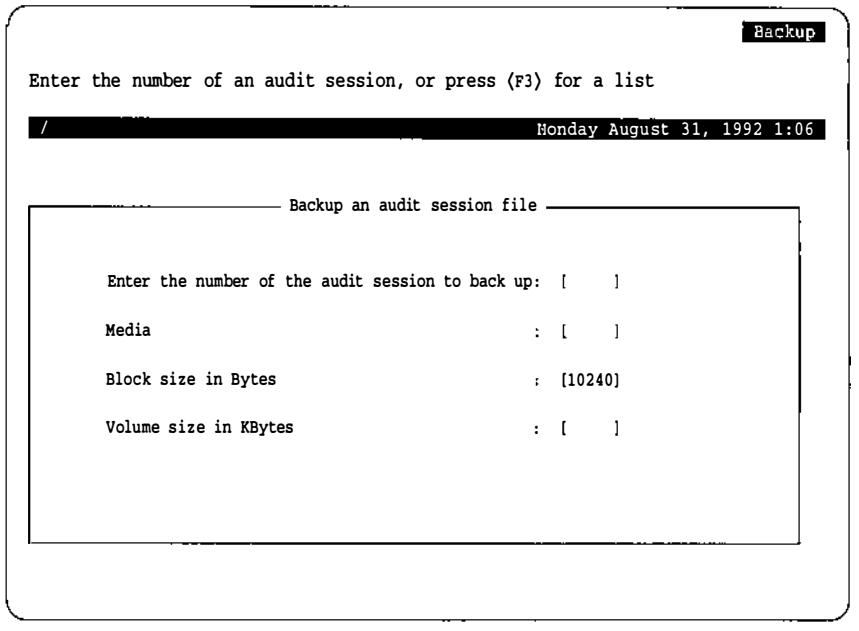
NOTE The session display is not a listing of session data files. It is the total number of records and bytes of that session; the actual data is stored in a series of collection files.

Backing up audit files

To back up audit files, make the following **sysadmsh** selection:

System ➪ Audit ➪ Files ➪ Backup

A screen similar to the following is displayed:



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. The Backup option requires a session number as input. This can be obtained by pressing ⟨F3⟩ for a list of sessions. After selecting Backup, you must select an output device for the backup. This can be any removable media available on the system.

WARNING 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

Similarly, sessions that were backed up onto removable media using the interface program can be reloaded using the Restore option. To do so, insert the media containing the saved session files into the restore device, and specify the device name.

To back up audit files, make the following **sysadmsh** selection:

System ➪ Audit ➪ Files ➪ Restore

You are asked to enter the device to restore from; press ⟨F3⟩ for a list.

Removing audit files

To remove audit files after backing them up, make the following **sysadmsh** selection:

System ➪ Audit ➪ Files ➪ Delete

The Delete selection is provided for the removal of audit sessions. Sessions can be archived to backup media and removed to make room on the filesystem for more audit files. Sessions are removed using the session number, which can be obtained by pressing ⟨F3⟩. The session number is then presented to the Delete option to delete all of the files associated with that session.

Monitoring disk space consumption

You can monitor the disk space consumed by the audit records using the **sysadmsh** selection:

System ➪ Audit ➪ 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

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 contention for disk resources. Also, auditing requires large amounts of space, even with compaction. The subsystem warns you

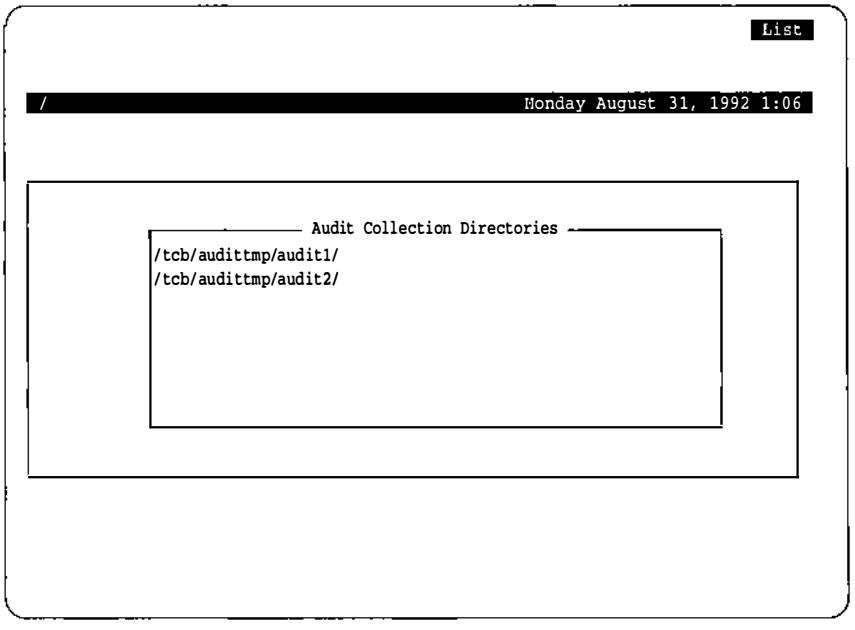
when disk space is low, and it eventually disables auditing if the free space of a filesystem is too low. For this reason, multiple directories are supported by the subsystem and the daemon. If an error occurs in writing to a directory or if space is exhausted, the subsystem and the daemon attempt to use alternate directories to continue.

Listing collection directories

To list the current audit collection directories, make the following **sysadmsh** selection:

System ➪ Audit ➪ Collection ➪ Directories ➪ List

A screen similar to the following is displayed:

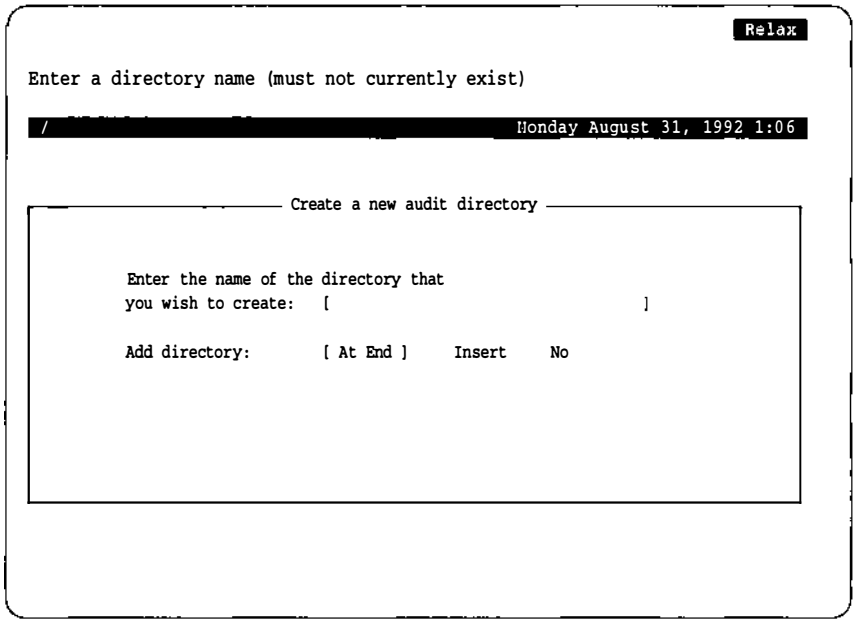


Creating a collection directory

To create a new collection directory (and optionally add it to the list used by the audit subsystem) make the following **sysadmsh** selection:

System ➪ Audit ➪ Collection ➪ Directories ➪ Create

A screen similar to the following is displayed:



You must supply the full pathname of the directory. The Add directory options are as follows:

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

Enter each filename as an absolute pathname. There is no artificial limit on the number of directories you may specify. If no directories are specified, the subsystem and the daemon create all files in the root filesystem using the reserved audit subsystem directory */tcb/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

To delete an existing collection directory, make the following **sysadmsh** selection:

System ➪ Audit ➪ Collection ➪ Directories ➪ Delete

You are asked to select the directory to be deleted. Press ⟨F3⟩ for a list.

Adding a collection directory entry

You can also add an existing directory to the list used by the audit subsystem. Directories are used in the order listed. Thus, a new entry can either be inserted into the list or placed at the end. To add an entry to the collection directory list, make the following **sysadmsh** selection:

System ➪ Audit ➪ Collection ➪ Directories ➪ Add

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

To remove an entry from the collection directory list, make the following **sysadmsh** selection:

System ➪ Audit ➪ Collection ➪ Directories ➪ Remove

You are asked to select the entry to be removed. Press ⟨F3⟩ for a list.

Report generation

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. Thus, 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.

The following Report options are available:

List	lists all report templates available
View	views the parameters stored in a report template
Create	creates a new report template
Modify	modifies an existing report template
Delete	deletes an existing report template
Generate	makes a reduction run, specifying audit session and report template

As discussed previously, audit collection criteria represents the first level of audit selection. After the data is gathered, it can be further processed, or reduced, to generate a useful collection of data about a specific aspect of system operation. The data reduction menus let you select to reduce and determine what records are desired. The Generate option supports a wide range of post-selection criteria that helps you target specific events, users, or objects. This requires the session number and the report template, which may be any of the report templates built using the report template create or update options.

The options for List, View, Create, Modify, and Delete are used for report template maintenance.

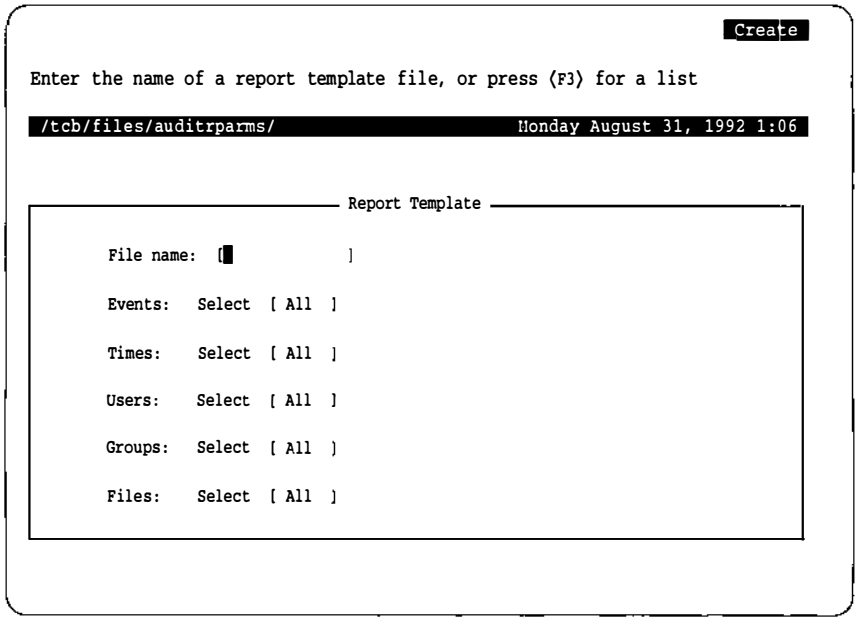
Creating or modifying a report template

To create or modify a report template, make one of the following **sysadmsh** selections:

System ➪ Audit ➪ Report ➪ Create
System ➪ Audit ➪ Report ➪ Modify

The screens for the Create and Modify options are identical. Modify allows you to take an existing report template and make any modifications desired.

A screen similar to the following is displayed:



Enter the name of the template file to be created or modified. Use ⟨F3⟩ to select from a list of templates; several are distributed with the system.

The selections are as follows:

Events — The audit events can be selected or all collected. Events not selected cause those records to be discarded from the output.

Times — The start and stop times for collection. If a security-related event was suspected between certain times of the day, you could use this feature to select those records that were generated during that time period. This could serve to concentrate the analysis on those records that are likely to reveal what has happened.

Users/Groups — Both users and groups of users can be singled out for audit. If a certain user account was the target of a penetration, you could select only those records that were generated from user or group IDs that matched that user. This permits the record search to be concentrated on suspected accounts.

Files — Files (object names) can also be used to select audit records from the output. For records that contain multiple object names, if a specified name matches *any* object in the record, the record is selected. The object names must be specified as absolute pathnames because all object names are resolved from relative to absolute names by the reduction program.

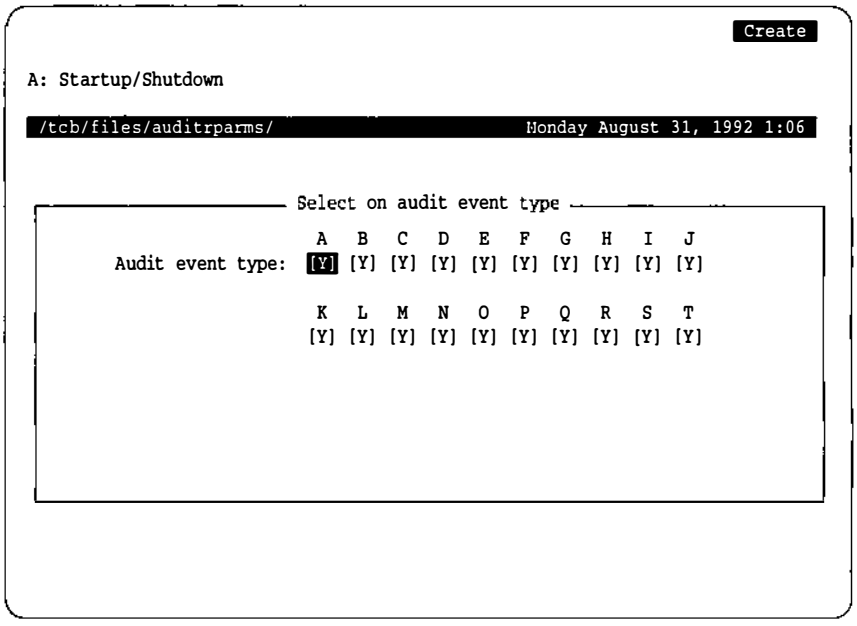
If All is chosen for a category, all events, times, and so forth are selected. If you choose Select, a window or form is displayed so that the desired criteria can be selected individually.

Any combination of the above criteria can be used. For instance, 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. Likewise, if time stamp selection is enabled and the record does not meet the criteria, the record is not selected. If the record passes the selection criteria for event type and time, then the record is selected if it has a user ID (login, effective, or real), group ID (effective or real), or an object in the record that is specified in the report template. If no users, groups, and objects are specified, only event type and time selection is performed.

Events

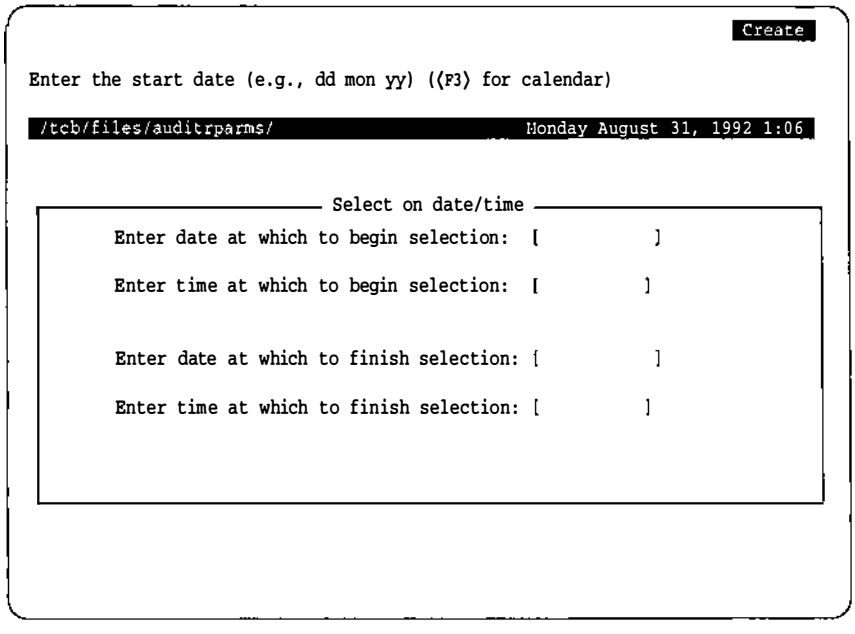
The selection for Events is highlighted first. If you choose Select and press ⟨Return⟩, the following window opens to allow the events to be selected:



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 the ⟨Space⟩. When you are satisfied, press ⟨Return⟩ to save your changes.

Times

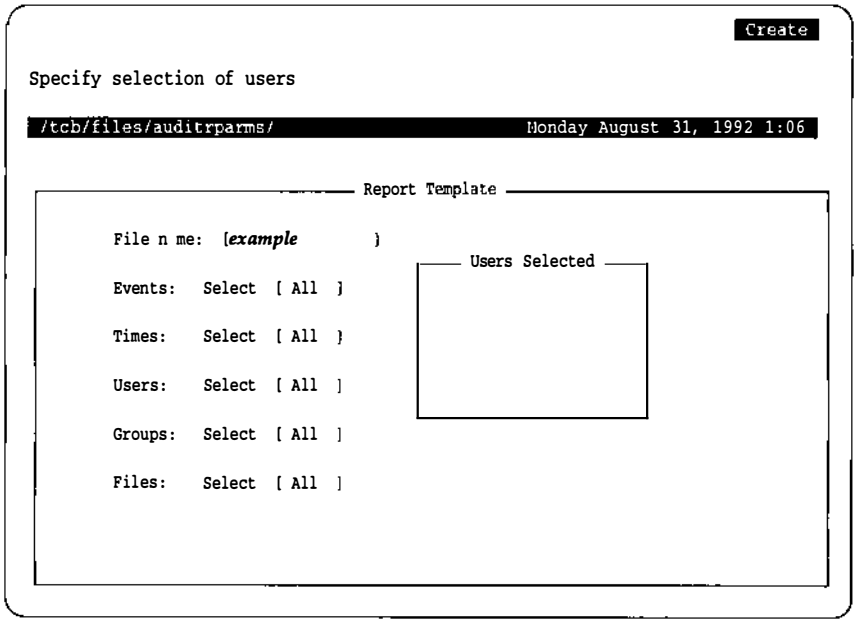
The Times selection is highlighted. Use the arrow key to highlight All, or Select and press ⟨Return⟩. If you choose Select, the following window opens to allow start and stop times to be selected:



You are asked in turn to enter the date and time for beginning and ending selection. You can press ⟨F3⟩ to get a calendar.

Users

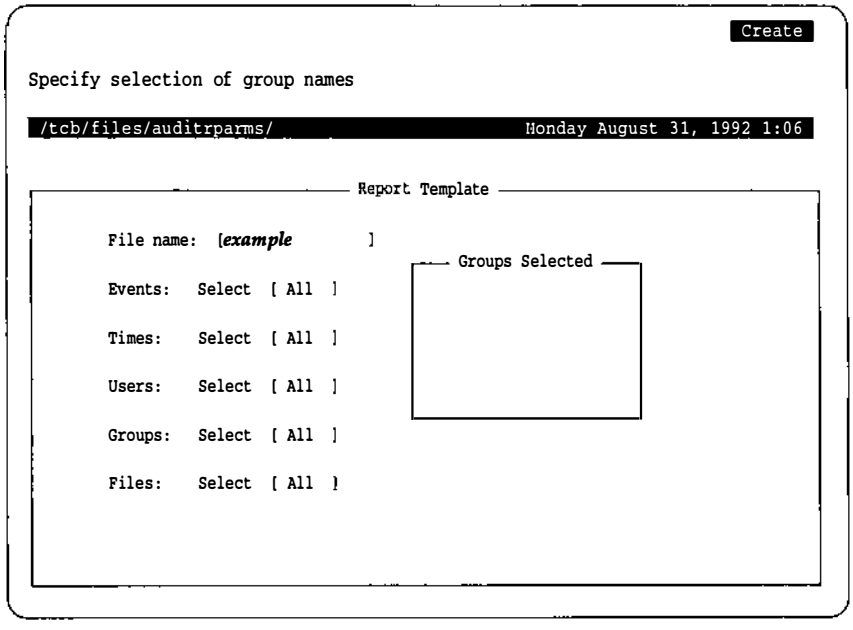
The Users selection is highlighted. Use the arrow key to highlight All, or Select and press ⟨Return⟩. If you choose Select, the following window opens to allow users to be selected:



You can modify the list of users and/or add new names to the list in the window. Press ⟨Return⟩ when the list is complete.

Groups

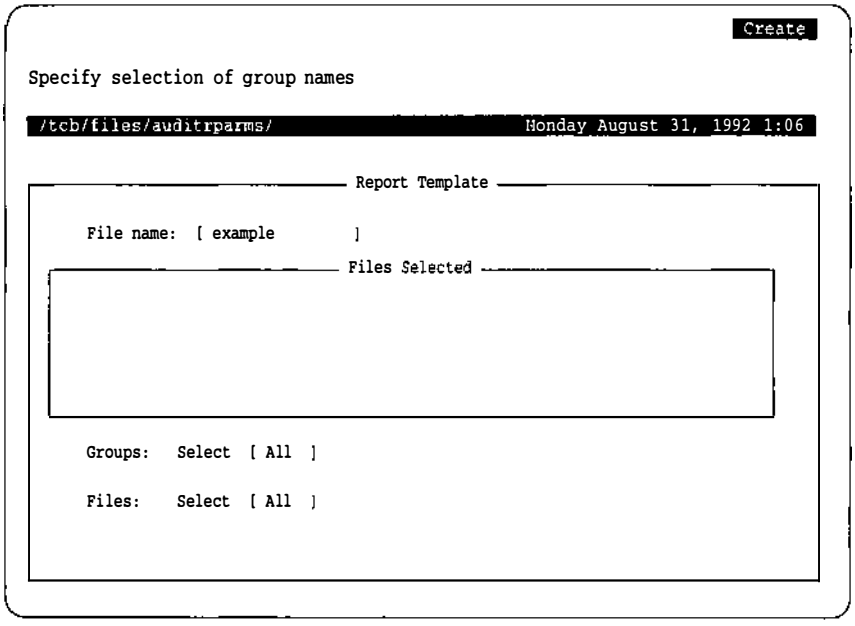
The selection for Groups is highlighted. Use the arrow key to highlight either All, or Select and press ⟨Return⟩. If you choose Select, the following window opens to allow groups to be selected:



You can modify the list of groups and/or add new names to the list in the window. Press ⟨Return⟩ when the list is complete.

Files

The final category is Files. If you specify the files (objects) here, it restricts the auditing of object deletion, modification, and so on to the files selected here. Use the arrow key to highlight either All, or Select, and press ⟨Return⟩. If you choose Select, the following window opens to allow files to be selected:



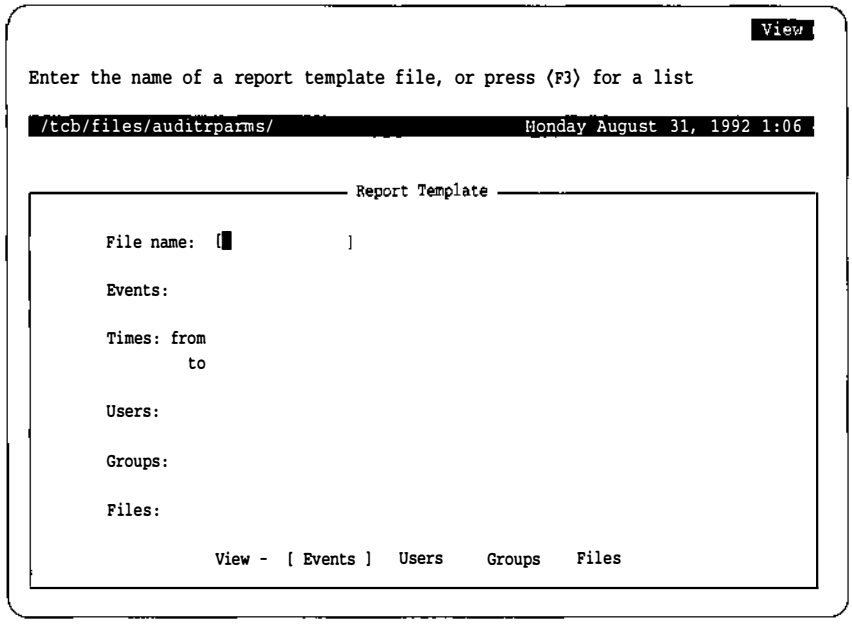
You can modify the list of files and/or add new names to the list in the window. Press ⟨Return⟩ when the list is complete.

Viewing a report template

To view a report template, make the following **sysadmsh** selection:

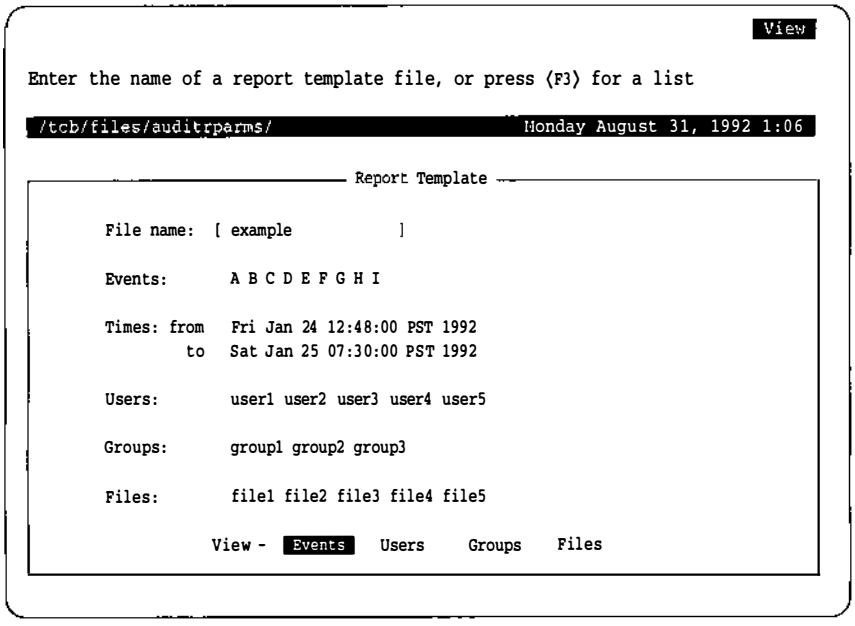
System ➪ Audit ➪ Report ➪ View

A screen similar to the following is displayed:



Use the ⟨F3⟩ key to select a template to view. You cannot make changes through this selection; if you wish to customize a template, you should use the Modify selection instead.

Once a template is selected, the fields are filled in as in the following example:



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

To list the available report templates, make the following **sysadmsh** selection:

System ➪ Audit ➪ Report ➪ List

A list of the report templates available is displayed. The following are shipped with the system:

admin.actions These are administrator actions; this is event type L.

all.objects These are actions relating to the creation, modification, or removal of objects; these are event types D-I.

authorization This is use of authorization; these are event types M and T.

dac.events These are DAC (Discretionary Access Control) changes or denials; these are event types J and K.

denials These are DAC denials; these are event types A-K, and N.

login.action This is a record of logins (successful and unsuccessful) and logouts; this is event type B.

Removing report templates

To remove a report template, make the following **sysadmsh** selection:

System ➪ Audit ➪ Report ➪ Delete

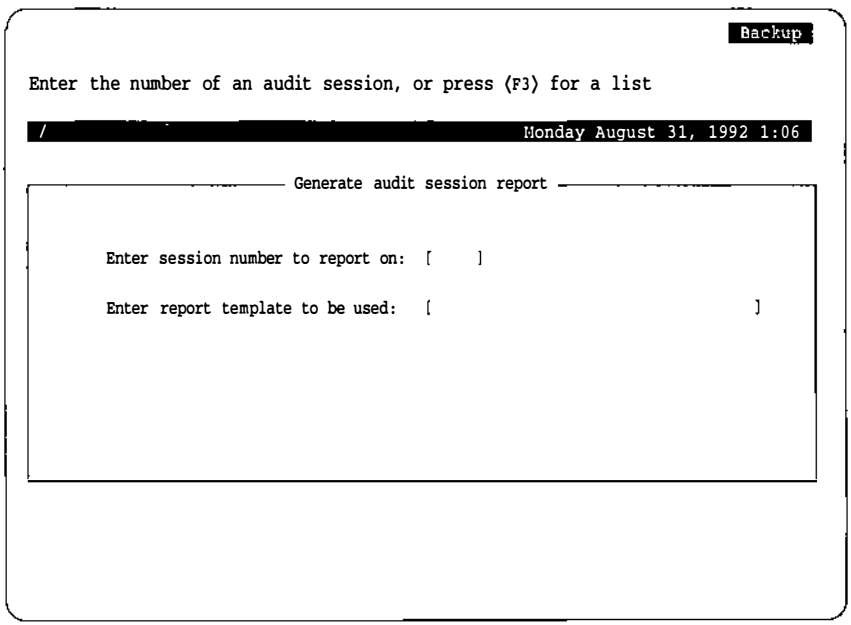
You are asked to provide the name of the report template to remove. Press ⟨F3⟩ for a list.

Generating an audit report

To generate an audit report, make the following **sysadmsh** selection:

System ➪ Audit ➪ Report ➪ Generate

A screen similar to the following is displayed:



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 10-3 is an example of an audit report based on a template with the following 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 278) contains a detailed study of how audit information is interpreted.

Example 10-3 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
```

(Continued on next page)

(Continued)

```
                ***** 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 instance, a process creation, **fork**(S), need only indicate the process ID of the newly created process and the ID of its spawning process (parent). However, for an **open**(S) system call, an object is being acted upon and the name of that object must be recorded. For system calls like **mount**(S) and **link**(S), still more information must be recorded; each requires that two object names be recorded. The reduction 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 instance, the date and time of the record and the process ID associated with the record are printed for each type. Beyond this, the content of a record depends on what was audited.

System call record formats

System call records account for the majority of the records in the audit trail. The operating system contains over 60 system calls. Not all of these system calls are audited as only some of these are deemed to be security-related.

Slightly over half of the system calls have the potential to create an audit record. Some system calls support multiple functions (such as **fcntl**(S), **msgsys**(S), **shmsys**(S), and **semsys**(S)) that may only generate audit records for certain functions. For instance, the **fcntl**(S) system call allows files to be opened by duplicating open file descriptors and also permits file-specific flags maintained by the kernel to be retrieved. The first case constitutes an auditable event, making an object available to a subject, while the second has no real security relevance. Furthermore, system calls may perform functions that are considered auditable events but are not enabled by the system event mask at the time.

For the most part, the output of system call records is the same for all calls. Variations exist because some system calls operate on objects (such as **open**(S)) and the object name is contained in the record. Each contains at least the time, date, process ID, system call name, event type, login user ID, real user and group IDs, effective user and group IDs, and an indicator of success or failure for the call.

Each output record contains these basic information fields and others depending on the system call. The basic record is shown in Example 10-4. This illustrates the common header along with the system call and result fields.

Example 10-4 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, **msgsys**(S) is the system call entry for message queue IPC operations. This single entry point calls **msgget**(S), **msgop**(S), and **msgctl**(S) to perform certain IPC functions.

System calls like this are not self-explanatory. The audit subsystem is aware of these overloaded calls and provides additional information to identify the particular function. For system calls that succeed, the result is specified as successful. For each that returns an error, the error provides additional record classification. For instance, an **open**(S) that fails from lack of permission is classified as an access denial. An unsuccessful system call that generates an audit record indicates the error in the result field.

The system call output records can be divided into two groups. The first group contains records that do not require pathnames in the audit record. For instance, the **fork**(S) system call is audited to track new processes as they are spawned into the system, but the audit record does not require a pathname. On the other hand, **open**(S) returns a file descriptor for the specified pathname. Subsequent operations, like **close**(S), use the file descriptor. To provide meaningful audit records, this second type of record must contain the pathname. Using the reduction function, this pathname is associated with all further actions on that file, even though the action may have been performed with a file descriptor.

Figure 10-1 lists audited system calls that do not contain pathname information.

pipe	fork	kill	close
setuid	setgid	exit	security
read	setpgrp	msg	dup
sem	shm	write	fcntl

Figure 10-1 System calls without pathnames

An output record from one of the above system calls uses the generic record mask described in Example 10-4. The following example illustrates the output record from a successful **setuid**(S) system call.

Example 10-5 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 10-6 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 10-6 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 **security**(S) system calls are overloaded. They map to multiple functions. These audit records identify the specific function being performed

and also the affected object (for example, shared 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 Figure 10-1, 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 due to **exec**(S), **exece**(S), **close**(S), or **exit**(S), the name of the object and an indicator of whether the file was read or written is included in the system call record for the action that caused the file to be closed. This is illustrated in Example 10-7.

Example 10-7 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 Figure 10-2, contains pathnames as part of the output record. The pathname represents the target of the system call. Two of the system call records actually contain two pathnames: **link**(S) and **mount**(S).

open	unlink	creat
exec	chdir	mknod
chown	chmod	stat
umount	exece	chroot
link	mount	

Figure 10-2 System calls with pathnames

Each of the system calls in Figure 10-2 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 10-8 shows an output record generated from a **creat**(S) system call. The record format is the basic format augmented by the pathname.

Example 10-8 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 10-9.

Example 10-9 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 10-10 illustrates the output record from a **chmod**(S) system call.

Example 10-10 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
- user password events
- protected database events
- audit subsystem events
- authorized subsystem events
- terminal and user account lock events

Each record contains some information common to all audit output records. This includes the process ID, the time and date, and the audit event type. The remainder of the output record depends on the record type. The record-specific fields are described in the following sections.

Login/Logoff record

All attempts to log into the system are audited by the **login** program. This is true of successful as well as unsuccessful attempts. This creates an important trail of user accesses to the system and also a trail of attempted accesses. You can use the audit records for login or logoff to determine who actually used the system. It is also valuable in determining if repeated penetration attempts are being made. The operating system supports the option of locking terminals after a certain number of attempts and this event can also be audited. Thus, you have all tools necessary to monitor (and prevent) access to the system.

Each login record contains an indicator of the specific action that was audited. The three possibilities are: successful login, unsuccessful login, or logoff. All successful logins and logoffs result in an audit output record that indicates the user account and terminal of the 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. Example 10-11 illustrates the output from a successful login.

Example 10-11 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 10-12 shows an audit record for an unsuccessful password change.

Example 10-12 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. See Example 10-13.

Example 10-13 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 **sysadmsh** audit selections and the audit daemon, **auditd**, both generate audit records for functions they support. Additionally, the audit device driver also writes audit records for certain function requests. The functions audited include the following:

- subsystem initialization
- subsystem termination
- subsystem parameter modification

- audit daemon enabled
- audit daemon disabled
- subsystem shutdown
- subsystem error

Each output record includes the common header information along with an indicator of the function audited. This provides an accurate accounting of all attempts to affect the operation of the audit subsystem. Example 10-14 shows an actual audit record written to indicate the startup and initialization of the subsystem.

Example 10-14 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 instance, if a subsystem encounters permission problems with a file or is denied service due to lack of memory or some other resource, the subsystem generates an error message to that effect. You can use these records to help maintain the security 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 10-15 shows a subsystem-generated audit record.

Example 10-15 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 instance, if a terminal is used to enter the system and the result is a series of unsuccessful logins, the **login** program may lock the terminal after a specified number of tries. Similarly, if a user attempts to 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. 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 10-16 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
```

Auditing capabilities extended 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 subsystem authorization permits access to a subset of audit functions under System ➪ Audit ➪ Report. 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 user authorizations" (page 74) for information on assignment of the **audittrail** authorization.

Glossary of audit terms

audit collection file	A file written by the audit subsystem device driver containing the raw audit data from all audit sources on the system, including system calls, trusted applications, and authorized subsystems.
audit compaction file	A file written by the audit daemon containing buffers of data read from the audit device driver. The data can be in either a compacted or non-compacted format, depending upon options selected at the time the audit session was started.
audit daemon	A daemon process started when the system makes the transition to multiuser state. It reads the audit subsystem device to retrieve audit records, compacts these records, and writes them to a permanent compaction file for later reduction. The daemon also acts as an interface program that permits non-protected subsystems to write audit records to the audit device.

audit session	The period of time from audit enable until audit disable. During this time, the audit data is stored in compaction files written by the daemon. Each session is uniquely numbered and each file that is part of the session contains this unique ID in the filename. A master file is used for each session to collect session information and session file names for later reduction.
audit subsystem	Consists of the components that provide the trusted audit services. This includes the audit device driver, the kernel audit mechanism, the audit daemon, the audit administrator interface, and the audit reduction facility.
audit trail	The collection of audit data records from an audit session that can be reduced into a report of system activity.
audit reduction	The transformation of raw audit trail data into output records containing dates, user IDs, filenames, and event types. The output record describes the audited event in a readable text form.
configaudit	The kernel authorization that allows the audit parameters to be set for all users of the system.
event control mask	The user-specific mask maintained in the Protected Password database on a per-user basis. This mask controls whether the user event mask prevails over the system default event mask when auditing is enabled. Each bit set in the control mask causes the event disposition mask to take precedence.
event disposition mask	The user-specific mask used in conjunction with the event control mask for user audit event control. If the user event control mask has a bit set on, the corresponding bit entry in the event disposition mask determines whether the event is always audited or never audited. This holds true regardless of the system default event mask value.
event type	A classification for each audit record. Security-related events on the system are classified into certain types that can be used to control audit generation or reduction. Every system action, regardless of success or failure, can be classified into an event type. This event type then determines the disposition of the record.

object	An entity acted upon by a subject (such as files, shared memory segments, semaphores, pipes, or message queues).
post-selection	The selective use of collected audit data. Post-selection involves collecting audit data for all events and users so the audit trail is as complete as possible. Any security-related event is in the audit trail compaction files at the end of a session.
preselection	Selectively controls audit record generation. This allows certain users and events to generate audit records while others are discarded. The result is a more compact audit trail with less detail than if full auditing was used.
report templates	Are generated through the administrative interface to control the selective reduction of audit sessions. Selective criteria control the user, object, and event selection for output records.
subject	An active entity that performs actions on objects, such as a process on the system that accesses files.
suspendaudit	A kernel authorization that suspends auditing.
system audit mask	The default system event mask used to determine what events are audited when a user process mask does not take precedence.
user audit mask	Collectively refers to the event control and event disposition masks that, together with the system default mask, control the generation of audit records on a per-process basis.
writeaudit	A kernel authorization that allows specific information to be recorded by the audit trail.

Chapter 11

Administering serial terminals

The most important aspect of a multiuser system is the addition and maintenance of serial terminal devices. Adding terminals lets more users access the system and adds to overall system capabilities.

This chapter explains the following tasks:

- using and maintaining serial terminals
- configuring and using **mscreen**(M), a serial terminal equivalent of **multiscreen**(M)
- using scancode-compatible terminals

Changing default terminal line characteristics

Your system can automatically adapt to several different terminal baud rates and settings. The same program that displays the login message, **getty**(M) (for "get tty"), reads these terminal line values from a table, trying each setting until one is successful, and the user can log in to the system. This table provides several default settings for different kinds of terminal lines.

getty automatically executes as part of the login process. The table of terminal settings is found in a file called */etc/gettydefs*. You can edit *gettydefs* to add different sets of terminal characteristics or to change the existing ones.

The gettydefs file

The file */etc/gettydefs* contains the information that **getty** uses to set up terminal line characteristics such as baud rate. The file is in the form of a table. Each table entry is divided into five fields and one optional field. These fields include:

label # initial-flags # final-flags # login-prompt # next-label

The fields are:

label identifies the *gettydefs* entry to **getty**. This could be a number or a letter. The label held corresponds to the line mode field in */etc/inittab*. **init** passes the line mode to **getty** as an argument.

initial-flags sets terminal line characteristics when **getty** first establishes the connection. **getty** recognizes the flags listed in the **termio**(M) manual pages. Often the only flag in this field is the one setting the baud rate. For example, B300 would set the speed to 300.

final-flags sets the terminal line characteristics just before **getty** executes **login**. These flags describe the operating characteristics for the line. The baud rate (B) is set again. Other common flags include **SANE** (a composite flag that sets a number of terminal characteristics to reasonable values), **TAB3** (expands tabs with spaces), **IXANY** (enables any character to restart output), and **HUPCL** (hangs up line on final close). Flags can be entered in any order.

login-prompt contains the login message that greets users. This field is printed exactly as it is entered, including spaces and tabs. An "**@**" symbol in the login-prompt field is replaced with the system node name (set by **uname**(C)).

Several character sequences are recognized, including:

\n	Line feed	\t	Tab
\r	Carriage return	\f	Form feed
\v	Vertical tab	\b	Backspace
\ *nnn*	(3 octal digits) The specified ASCII characters		

next-label identifies the next label in *gettydefs* for **getty** to try if the current one is not successful. **getty** tries the next label if a user presses the ⟨Break⟩ key while attempting to log in to the system. Groups of entries, such as dial-up or TTY lines, should form a closed set so that **getty** cycles back to the original entry if none of the entries is successful.

Each field is separated by a number sign (#), and each entry in *gettydefs* is separated by a blank line.

Here is the default entry from */etc/gettydefs* for terminal lines:

```
m # B9600 HUPCL # B9600 CS8 SANE HUPCL TAB3 ECHOE IXANY #\r\n@!login: # m
```

Here is a description of each part of this line:

- The letter "m" identifies this entry to **getty**.
- The next field sets the baud rate to 9600.
- The third field indicates the baud rate (B9600), and several line characteristics including **SANE** (a composite flag for a number of characteristics), and **HUPCL** (hangs up line on final close).
- "\r\n@!login:" is expanded to display the system name in the "login:" prompt.
- The last field is the ***next-label***. In this example, the label "m" directs **getty** to repeat this entry if it is unsuccessful. In the case of dialup lines, these labels can be used to direct **getty** to cycle through a series of lines (for example, 300-1200-2400).

Changing the gettydefs file

The file */etc/gettydefs* has sets of entries for the dial-up lines and terminal lines. These different sets correspond to line-mode settings in */etc/inittab*. The **init** program passes the line mode as an argument to **getty**.

You can edit *gettydefs* to add new terminal settings or to change existing ones. For example, the settings for terminal lines might look like the following:

```
4 # B2400 HUPCL # B2400 CS8 SANE HUPCL TAB3 ECHOE IXANY #\r\n@!login: # 5
5 # B4800 HUPCL # B4800 CS8 SANE HUPCL TAB3 ECHOE IXANY #\r\n@!login: # 6
6 # B9600 HUPCL # B9600 CS8 SANE HUPCL TAB3 ECHOE IXANY #\r\n@!login: # 4
```

To change the sample *gettydefs* file so that the first baud rate **getty** attempts is 1200, do the following:

1. Enter a text editor to edit the first line of the file *gettydefs*.
2. Change the first and third fields from B2400 to B1200.
3. Save *gettydefs* and exit the editor.

The sample file should look like the next example:

```
4 # B1200 HUPCL # B1200 CS8 SANE HUPCL TAB3 ECHOE IXANY #\r\n@|login:# 5
5 # B4800 HUPCL # B4800 CS8 SANE HUPCL TAB3 ECHOE IXANY #\r\n@|login:# 6
6 # B9600 HUPCL # B9600 CS8 SANE HUPCL TAB3 ECHOE IXANY #\r\n@!login:# 4
```

You can also add additional terminal line settings to *gettydefs*. Flags and permissible values for terminal settings are listed in the **stty**(C) manual page.

When you add a new entry, be sure that the groups of entries in *gettydefs* form a closed set, so the next-label field of the last entry directs **getty** back to the first entry in the group.

To add an entry for a baud rate of 300 to the preceding sample *gettydefs* file, follow these steps:

1. Enter a text editor to edit the file */etc/gettydefs*.
2. Locate the point where you want to insert the new settings for *gettydefs*. The order of the entries does not matter; **getty** only looks for the label. In this example, the new entry is the last entry in the file.
3. Insert a carriage return after the last line in the file and enter the following on a new line:

   ```
   7 # B300 HUPCL # B300 CS8 SANE HUPCL TAB3 ECHOE IXANY #\r\n@!login:# 4
   ```

4. To incorporate label 7 into the set of labels, change the next label field for entry 6 to 7:

   ```
   6 # B9600 HUPCL # B9600 CS8 SANE HUPCL TAB3 ECHOE IXANY #\r\n@!login:# 7
   ```

   getty is now directed from label 6 to 7, and then back to 4. Make certain that a blank line separates each pair of entries.
5. Exit the text editor, saving the revised *gettydefs* file.

The new *gettydefs* looks like the following:

```
4 # B1200 HUPCL # B1200 CS8 SANE HUPCL TAB3 ECHOE IXANY #\r\n@!login: # 5
5 # B4800 HUPCL # B4800 CS8 SANE HUPCL TAB3 ECHOE IXANY #\r\n@!login: # 6
6 # B9600 HUPCL # B9600 CS8 SANE HUPCL TAB3 ECHOE IXANY #\r\n@!login: # 7
7 # B300 HUPCL # B300 CS8 SANE HUPCL TAB3 ECHOE IXANY #\r\n@!login: # 4
```

Checking the terminal settings

Each time you change the terminal line settings or add new entries to *gettydefs*, you should check to make sure that the new values make sense to **getty**. To do this, you use the command **getty** with the check option, **-c**, and the filename.

For example, to check *gettydefs*, enter:

getty -c /etc/gettydefs

The file is scanned and the results are displayed. If any of the values and settings in *gettydefs* are not permitted, **getty -c** reports them. For more information on **getty** and *gettydefs*, see the **getty**(M) and **gettydefs**(F) manual pages.

Changing serial line operation

Whenever you enable a terminal with the **enable** command, the system automatically sets the operating characteristics of the serial line to a set of default values. Sometimes these values do not match the values used by the terminal and, therefore, must be changed to allow communication between the system and the terminal. You can display the operating characteristics of a serial line with the **stty** (for "set tty") command. If you need to change the characteristics of a port that is enabled, you should use the entires in the *gettydefs* file rather than the **stty** commands given below.

NOTE Any settings on a port using the **stty** command only last as long as the port in question is still open. As an example, if you want to change the baud rate of tty2a, and tty2a is not enabled, the **stty** command first opens the port, then changes the port settings, and finally closes the port. When it closes the port for the last time, the settings revert to the original. In the **stty** commands later in this section, the use of the **while** loop is to avoid this behavior of **stty**. If you run **stty** redirecting input without "**< /dev/*ttyname***", it works on your current serial line, which you have open. In this case, because the serial line stays open after the **stty** command, the settings also stay in place.

You can display the current operating characteristics of a serial line by entering this command at the terminal connected to that line:

stty -a

If it is impossible to log in at that terminal, you can use another terminal to display the characteristics. Log in as the super user at another terminal, and enter:

stty < *ttyname*

where ***ttyname*** is the name of the device special file corresponding to the serial line (see the "UNIX directories and special device files" chapter in this guide). For example, this command displays the current characteristics of the serial line named */dev/tty1a*:

stty < /dev/tty1a

The command displays the baud rate, the parity scheme, and other information about the serial line. This information is explained in the **stty**(C) manual pages.

One common change to a serial line is changing the baud rate. This is usually done from a terminal connected to another serial line because changing the rate disrupts communication between the terminal and the system. Before you can change the rate, you need to know the current baud rate of the

terminal (review the terminal hardware manual to see how to determine the current baud rate). Once you have the baud rate, log in as the super user at the other terminal, and enter:

(stty *baud-rate*; while : ; do sleep 3600; done) < *ttyname* &

where ***baud-rate*** is the current baud rate of the terminal, and ***ttyname*** is the name of the device special file corresponding to the serial line you wish to change. The baud rate must be in the set 50, 75, 110, 134, 150, 200, 300, 600, 1200, 2400, 4800, and 9600. For example:

(stty 9600; while : ; do sleep 3600; done) < /dev/tty1a &

This command changes the baud rate of the serial line */dev/tty1a* to 9600. Note that the "less than" symbol (<) is used for both displaying and setting the serial line from another terminal.

Another common change is the way the system processes input and output through the serial line. Such changes are usually made from the terminal connected to the serial line. For example:

stty tabs

This command causes the system to expand tabs with spaces (used with terminals that do not expand tabs on their own). Another example is:

stty echoe

This command causes the system to remove a deleted character from the terminal screen when you back over it with the ⟨Bksp⟩ key.

Note that the **stty** command may also be used to adapt a serial line to an unusual terminal, to another type of serial device that requires parity generation and detection, or to special input and output processing.

For a full description of this command, see the **stty**(C) manual page.

Setting the terminal type

UNIX systems require that the terminal type be clearly defined before any work is done at the terminal. The preferable method for setting your terminal type is to assign the type to the **TERM** variable, a special environment variable that associates the terminal you are using with a list of characteristics given in the */etc/termcap* file. The characteristics tell the system how to interpret your terminal's keys and how to display data on your terminal screen.

If you are using the Bourne or Korn shell (**sh**(C) or **ksh**(C)), the **TERM** assignment has the form:

TERM=*termtype*; export TERM

If you are using the C shell (**csh**(C)), the **TERM** assignment has the form:

setenv TERM *termtype*

The ***termtype*** must be one of the names associated with one of the terminals defined in the */etc/termcap* file. The assignment must be entered at the terminal whose type you are setting.

For example, to set the terminal type to "ansi" from Bourne shell, go to the terminal you wish to set, enter at the "$" shell prompt:

TERM=ansi; export TERM

From the C shell, enter at the "%" shell prompt:

setenv TERM ansi

If you are not sure which name you may use for ***termtype***, you can view the names either by displaying the */etc/termcap* file, or by reading the **terminals**(M) manual page which lists all terminals supported in the */etc/termcap* file. To display the file itself, enter:

more /etc/termcap

You can let the system define the terminal type automatically whenever you log in by including the **TERM** assignment in your *.profile* file. (See "Modifying .profile and .login files" in the "Customizing system startup" appendix of this guide.)

For an alternate method of setting your terminal type, see the manual page for **tset**(C).

If you let the system set the terminal type, be careful when logging in on terminals that are not the same as your normal terminal. The system has no way of checking whether or not the terminal assignment is correct for the given terminal and assumes that it is the same as your normal terminal. If it is not, you must set the terminal type manually.

Setting the terminal type automatically

If you want to have the terminal type set automatically at login time, follow this procedure:

1. Log in on the terminal in question and determine which ***ttyname*** you are using by entering the **tty**(C) command:

   tty

2. Log in as *root* and edit the file */etc/ttytype* with a text editor. Change the terminal type field for the line associated with the terminal in question to the terminal type you desire to use. Follow the model for the console. If you want your terminal type to be set to "wy50" for */dev/tty1a*, edit */etc/ttytype* as follows:

   ```
   wy50 tty1a
   ```

3. Then the user's start up file must be edited with the appropriate **tset**(C) command line to set the terminal type automatically. In each C-shell user's *.login* file, add the following line:

   ```
   tset -s -Q > /tmp/tset$$; source /tmp/tset$$; /bin/rm /tmp/tset$$
   ```

   Be sure to remove the default **setenv**(C) command line involving **TERM** and **TERMCAP** from the *.login* file.

   In each Bourne shell or Korn shell user's *.profile*, add the following line:

   ```
   eval `tset -s`
   ```

   Be sure to remove the existing **tset** command line from the *.profile* file.

4. Have all users log out, then log in again to test the new terminal type change. After they log in, have them verify the new term type by entering the **env**(C) command:

   env

Removing a terminal

From time to time it may be necessary to remove a terminal from the system, for example, if you wish to replace it with some other device. Before you can remove a terminal, you must disable it with the **disable**(C) command.

To remove a terminal, follow these steps:

1. Turn off the power to the terminal.
2. Log in as the super user at another terminal.

3. Use the **disable** command to disable the terminal. The command has the form:

   disable ***ttyname***

   where ***ttyname*** is the name of the serial line to which the terminal is attached. For example:

   disable tty1a

   This command disables the terminal connected to serial line */dev/tty1a*.

4. Disconnect the terminal from the system.

The serial line previously connected to the terminal is now free to accept another device.

Setting up a serial console

You can configure a serial device, rather than a display adapter, as your system console. The **boot**(HW) program sets the default console at boot time according to the following procedure:

A. The **boot** program looks for the entry **SYSTTY=***x* (where *x* is a number that specifies the system console device) in the */etc/default/boot* file.

B. If the **SYSTTY** entry is not found or the */etc/default/boot* file is not readable, **boot** checks your system for a display adapter and designates it as your system console.

C. If no display adapter is found, **boot** looks for tty1a, sets the serial port to 9600 baud, 8 data bits, 1 stop bit, and no parity, and uses it as the system console.

To change the console device to a serial terminal, do the following:

1. Enter system maintenance mode.

2. Enter the following commands:

   enable tty1a
   disable tty0[1-9] tty1[1-2]

   This enables the serial device and disables the console multiscreen devices associated with a display adapter.

3. Edit the */etc/default/login* file. In this file you should change the value of the parameter **OVERRIDE** from "tty01" to the tty line of the serial console; normally that value is "tty1a". After editing, the line in */etc/default/login* (if the tty line is *tty1a*), should read:

   ```
   OVERRIDE=tty1a
   ```

4. Edit the file */etc/default/boot* and add the following line to the end of the file:
   ```
   SYSTTY=1
   ```
5. After you reboot, the system recognizes the new console device.

To temporarily change the system console device from the command line, enter **systty=*x*** at the boot prompt (where *x* is "cn" for a display adapter or "sio" for a COM1 serial port). This does not create or change a **SYSTTY** entry in the */etc/default/boot* file.

Using serial multiscreens with mscreen

If you are familiar with **multiscreen**(M), the feature that 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. This means that 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 focuses on the Wyse 60, using its two pages of screen memory as the basis for all examples. (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 pseudo-ttys" (page 299) 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, all of whom use two screens, could make your system 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. Note that 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.

As installed, the system does not have any pseudo-ttys. 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 and 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.

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. This is not to say that

other terminals do not work with **mscreen**; they do. You need only 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, check the following common problems:

- 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 uses

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 11-1 allows three full-featured mscreens on a Wyse 60 and adds a number of convenience features for the **mscreen** user. Example 11-2 presents the same material for the Bourne and Korn shell *.profile* file. Note that these examples are designed to be added to the end of your *.login* or *.profile* file, and replace any existing **tset** material.

Example 11-1 .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 11-2 .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 (that **mscreen** uses to store the contents of multiple screens) as part of the initialization string. In Examples 11-1 and 11-2, **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, during first login procedure, the string "\033eG\c" is echoed. 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 Examples 11-1 and 11-2.

Using scancode-compatible terminals

Most terminals send information to the operating system only in the form of keytop values, which are the characters that appear on the faces of the keys. However, a few terminals can also send PC scancodes, which are unique values associated with the depression and the release of each key. Several applications and environments now use PC scancodes and more are under development.

A scancode application running on a terminal that is in PC-scancode mode can access more distinct keystrokes than character mode would provide. For example, if you set your terminal to character mode and press the key labeled "A", your terminal sends a single value (the ASCII value of " a ") to your application. However, if you set your terminal to scancode mode and press the key labeled "A", your terminal sends one value when you depress the key and a second value when you release the key. A scancode application translates these scancode values according to a predetermined map.

Using function keys in scancode mode

When you set up your terminal and system to run in scancode mode, your function keys get set to their default values. If you want to program your function keys while you work in scancode mode, you must use the **setkey**(C) or **mapstr**(M) utility, rather than your terminal's setup procedure. The **setkey** command lets you program one key at a time, while **mapstr -f** reads a file

containing the assignment for all the function keys. These utilities formerly affected only the console.

NOTE **scanon** does not run **mapstr**, so if you use **scanon** you also have to run **mapstr** to use the function keys, numeric pad and arrow keys.

The syntax for the **setkey** command is:

setkey ***keynum string***

The **setkey** command assigns the specified ANSI ***string*** to be the output of the function key ***keynum***. For example, for function key 1 (⟨F1⟩) to output the string "date", use this command:

setkey 1 "date"

For a key assignment to last beyond the current login session, place the **setkey** command in your *.login* file.

The syntax for the **mapstr** command is:

mapstr [-d] [*datafile*] [-f] [*termtype*]

Without the **-d** option, **mapstr -f** reads the function key values from the file in */usr/lib/keyboard/strings.d* that corresponds to the terminal type. To customize your function key assignments, create a new file for **mapstr** to read, using a file from */usr/lib/keyboard/strings.d* as a template. Then specify your new file in the **mapstr** command as follows:

mapstr -d *newfile* -f

For these key assignments to last beyond the current login session, place the **mapstr** command in your *.login* file.

Correcting a hung scancode-compatible terminal

If your PC-scancode application crashes, your terminal might hang with the terminal and the line discipline in incompatible modes. To correct this incompatibility, log into another terminal and use either the **scanon**(M) or **scanoff**(M) command as described below.

If you want to restore your terminal and line discipline to PC-scancode mode, enter the following **scanon** command, where ***ttyline*** is the tty of the hung terminal:

scanon /dev/*ttyline*

If you want to restore your terminal and line discipline to character mode, use the **scanoff** command:

scanoff /dev/*ttyline*

You do not need to be super user to use **scanon** and **scanoff** to affect your own tty. For more information on **scanon** and **scanoff**, see the **scanon**(M) manual page.

Chapter 12

Using modems

Modems (from *mo*dulate *dem*odulate) are a significant addition to your system, allowing you to communicate over phone lines from remote sites.

This chapter explains the following tasks:

- choosing a serial port
- installing and configuring modems
- dialing into your computer
- adding passwords for dial-in lines
- dialing out from your computer
- using and customizing dialers

Choosing a serial port

Note that physical connections between a device and the system vary according to hardware configuration. For specific information about connecting your serial device, refer to the hardware manuals provided with the device and with your computer. Before adding a modem, you should make certain you have a port available, either directly on COM1 or COM2, or from a multiport card.

Table 12-1 contains sample device names of serial ports with and without modem control.

Table 12-1 Serial ports

Device	Function
/dev/tty1a	main serial adapter without modem control
/dev/tty1A	main serial adapter with modem control
/dev/tty2a	alternate serial adapter without modem control
/dev/tty2A	alternate serial adapter with modem control

/dev/tty1a and */dev/tty1A* refer to the same serial port (likewise for */dev/tty2a* and */dev/tty2A*). The operating system uses different device-driver subroutines for each. Never attempt to use both modem and non-modem control ports at the same time or you will see the warning:

```
cannot open: device busy
```

For systems with non-intelligent multiport serial cards, */dev/tty*[*1,2*][*a-m*] are the non-modem control devices, and */dev/tty*[*1,2*][*A-M*] are the modem control devices.

Intelligent multiport serial cards are supplied with their own drivers and a have variety of device naming schemes; consult the documentation for your intelligent card for a list of device names.

For information on adding serial ports, see the *Hardware Configuration Guide.*

Configuring your modem

Proper modem configuration is necessary when using **cu**(C) and **uucp**(C). Modem settings differ for each modem. Consult your modem manual for the proper switch settings.

Smartmodem 1200 or compatible

If you have a Hayes Smartmodem 1200 or compatible, switches 3 and 8 should be down:

	1	2	3	4	5	6	7	8
up	•	•		•	•	•	•	
down			•					•

When switch 3 is down, the result codes are sent by the modem to the terminal or computer. When switch 8 is down, the modem can interpret the command being issued. This allows both MS-DOS and UNIX system communications systems to work.

Table 12-2 lists the functions of each switch setting.

Table 12-2 Hayes-Compatible switch settings

Switch	Position	Function
1	up*	Modem responds to DTR from computer
	down	Modem forces DTR high, so no signal is required from computer
2	up*	Result codes are in English
	down	Result codes are numeric
3	up	There are no result codes
	down*	Result codes are sent in response to each modem command
4	up*	Commands are echoed
	down	Commands are not echoed
5	up*	Modem answers phone
	down	Modem does not answer phone
6	up*	CD is asserted when carrier is actually present
	down	CD and DSR are forced high
7	up*	Modem is attached to single-line phone
	down	Modem is attached to multi-line phone
8	up	Modem does not recognize dialing commands
	down*	Modem recognizes dialing commands

The asterisks (*) indicate the switch settings required. If you have a different modem, consult your reference manual for the proper switch settings to both send and receive calls.

Smartmodem 2400, V-series 9600 or compatible

The Hayes 2400 and V-series 9600 Smartmodem or compatible modems are supported and are configured automatically when you use the dialer entries or dialer programs supplied with your SCO distribution. If you are using an unsupported dialer (one brought from elsewhere or written yourself), and you plan to use it for a dial-in line, be sure that you use the following settings:

AT&f fetches factory configuration.

ATT is tone dialing.

ATl0 is low speaker volume.

AT&d2 sets dtr "2": goes on hook when dtr drops.

AT&c1 sets dcd "1": dcd tracks remote carrier.

ATs0=1 answers phone after 1 ring (AA light should come on).

ATs2=128 disables modem escape sequence.

ATe0 does not echo (modem no longer echoes what is sent to it).

ATq1 is quiet mode (modem does not respond with "OK" after this command or any that follow).

AT&w saves settings in non-volatile memory. If you do not want to save the settings, you do not have to enter this command.

Telebit Trailblazer

If you have a Telebit Trailblazer modem or compatible, log in as *root* and enter the following command:

/usr/lib/uucp/dialTBIT -z /dev/tty*nn* 9600

where ***nn*** is the tty number of the serial line.

General modem configuration guidelines

When configuring non-supported modems, there are a number of things you should look out for:

- You may need to disable or reduce the level of data compression and error correction when using a modem for UUCP. If throughput seems too low, or you have failed transfers, try disabling these features.
- Do not enable XON/XOFF flow control when using the modem for UUCP. The stop characters (DC3) can occur in the UUCP protocol data stream, and thus cause transfer failures. Make sure that XON/XOFF flow control is turned off (XON/XOFF passthrough mode may be usable).

  You can use RTS/CTS flow control with your modem; see "RTS/CTS flow control" (page 320) for more information.

If you want to enable XON/XOFF when using the modem for **cu**(C), then you will need to have an alternate dialer defined. This can be done by modifying and renaming the Dialers file entry, or by modifying and renaming the dialer program, such that you have one dialer definition for UUCP and another for **cu**. See the **sysfiles**(F) manual page for instructions on how to do this.

Installing a modem

NOTE We strongly advise that you do not use an internal modem. We have successfully tested the Hayes 1200 and 2400 baud internal modems, and these modems are known to work. However, most Hayes-compatible internal modems have compatibility problems which prevent them from working properly under UNIX System V/386. These incompatibilities include:

- No support for quiet mode, no-echo mode, or auto-answer mode.
- Modem floods the bus with spurious interrupts.

When you are hooking up your modem, or any other device, make sure that serial wires connected to your computer are not left hanging. An unterminated line connected to your computer can considerably reduce system performance; always unplug a modem wire at the computer end instead of at the modem end.

Three-wire cables often used to connect terminals to the computer are not sufficient for connecting modems. For a modem cable on a 25-pin serial port, pins 2, 3, 7, 8, and 20 must be connected "straight-through" (meaning the pins are connected straight across: pin 2 to pin 2, pin 3 to pin 3, and so on). In addition, pins 4 and 5 must be connected straight through if RTS/CTS is used. If you are unsure as to what to use, a cable that connects all pins works correctly. Either a ribbon cable, or a straight-through cable will work. See **serial**(HW) for details of 9-pin connections.

To install your modem for dial-in or dial-out, follow these steps:

1. Make sure the UUCP package is installed. Use **custom**(ADM) to install it if necessary.
2. Make sure the serial port you have chosen for your modem is recognized at bootup (check */usr/adm/messages* or use **hwconfig**(C)) and, if the modem is internal, make sure that the COM port does not conflict with any other device. Only serial devices attached to COM1 and COM2 are supported by SCO.
3. Make sure the modem and non-modem control ports are disabled by using the **disable**(C) command. For example, the following commands applies to a modem connected to COM1:

   disable tty1a
   disable tty1A

4. If you are going to use the line for dial-out, the serial port must be owned by *uucp*. To make sure the line is owned by *uucp* enter these commands:

   chown uucp /dev/tty*nn*
   chgrp uucp /dev/tty*nn*

   where ***nn*** is the number of the corresponding line.

5. Connect the modem to the machine using a straight-through cable (pins 2 and 3 are not crossed). The cable must have at least pins 2, 3, 7, 8, and 20 connected. (This is for a 25-pin serial port. See **serial**(HW) for 9-pin connections.)

   Most standard COM ports use straight-through cables, but some hardware requires a null-modem cable (pins 2 and 3 are crossed). A standard COM port is known as DTE (Data Terminal Equipment); a port that needs a null-modem cable is known as DCE (Data Communications Equipment). Check your hardware documentation if you are unsure. If the COM board is a DCE, you need a null-modem cable.

6. If you intend to use the modem for dial-in, you need to check the */etc/inittab* file and confirm it has an entry for your port similar to the folllowing:

   ```
   Se1A:2:off:/etc/getty -t60 tty1A m
   ```

   /etc/inittab determines whether the port has a login prompt and defines the serial line characteristics. There should already be an entry for the line you are using; all you need to do is check the last field. This field is a number or letter (label) from the */etc/gettydefs* file, which associates a set of serial line characteristics (such as baud rate, parity, and number of stop bits). Use the label from that corresponds to the incoming baud rate you are using:

Baud rate	Label
1200	2
2400	3
9600	m or 6

   You can change */etc/inittab* with a text editor, but do not alter any fields other than the *gettydefs* label. For example, if you are using a 2400 baud modem on the first standard serial port, the line should be similar to the following:

   ```
   Se1A:2:off:/etc/getty -t60 tty1A 3
   ```

   The entry for a 9600 baud modem on COM2 should look like this:

   ```
   Se2A:2:off:/etc/getty -t60 tty2A m
   ```

   If you make any changes, use the following command to check the entries:

   grep -i tty*nn* /etc/inittab

   where **tty*nn*** is the tty you are using. This should generate only two lines: one for the modem control port (such as *tty1A*) and one for the non-modem control port (such as *tty1a*).

   For more information on the */etc/inittab* file and the various control codes, see the **getty**(M) and **inittab**(F) manual pages.

7. Next, you must duplicate any changes you made to the */etc/inittab* file in the */etc/conf/init.d/sio* file.

   WARNING Each time the kernel is relinked (as when a driver is added or a tunable parameter is changed), */etc/inittab* is reconstructed from the entries found in */etc/conf/init.d/sio*.

8. Add the correct entries to the */usr/lib/uucp/Devices* file. This file should have two entries for each serial port being used for a modem. One of the entries is used when you start a call using the modem (the Automatic Calling Unit (ACU) line), and the other line is used to connect directly with the modem to issue commands manually (the Direct line). For a Hayes-compatible modem operating at 2400 baud, using COM1, the entries in */usr/lib/uucp/Devices* should be:

```
ACU  tty1A - 1200-2400 /usr/lib/uucp/dialHA24 \T
Direct  tty1a - 1200-2400 direct
```

Make sure that the entries do not have a number sign (#) in front of them. This is the syntax to show that the line is only a comment, and is to be ignored. There are many examples in the *Devices* file that are commented out with this character.

WARNING Be sure that any "Direct" entries in the *Devices* file appear *after* the corresponding ACU entries. If this is not so, the modem may not be reinitialized properly after an outgoing call.

9. Test the modem's ability to dial correctly. (This process is outlined in Figure 12-1.) For example, the following command uses *tty1a*:

cu -ltty1a dir

You should see a message indicating that you are connected. If you see the message "cu: dir permission denied," the user executing the **cu** command does not have write permission on the */usr/lib/uucp/Devices* file.

If you do not see such a message, and there was no message to indicate that you connected correctly, check the following:

- ensure the "Direct" entry in the *Devices* file is correct
- make certain you entered the **cu** command correctly
- run the **hwconfig** utility and make sure your serial port is configured
- check the permissions on the port (see step 4)

NOTE The instructions that follow assume a Hayes-compatible command set and response codes. Other modems may use other conventions. Consult your modem documentation for further details.

10. If you see a message confirming your connection, enter the following on your keyboard:

AT

The screen should echo "OK". If the modem is set to return result codes as numeric codes rather than text, you see "0".

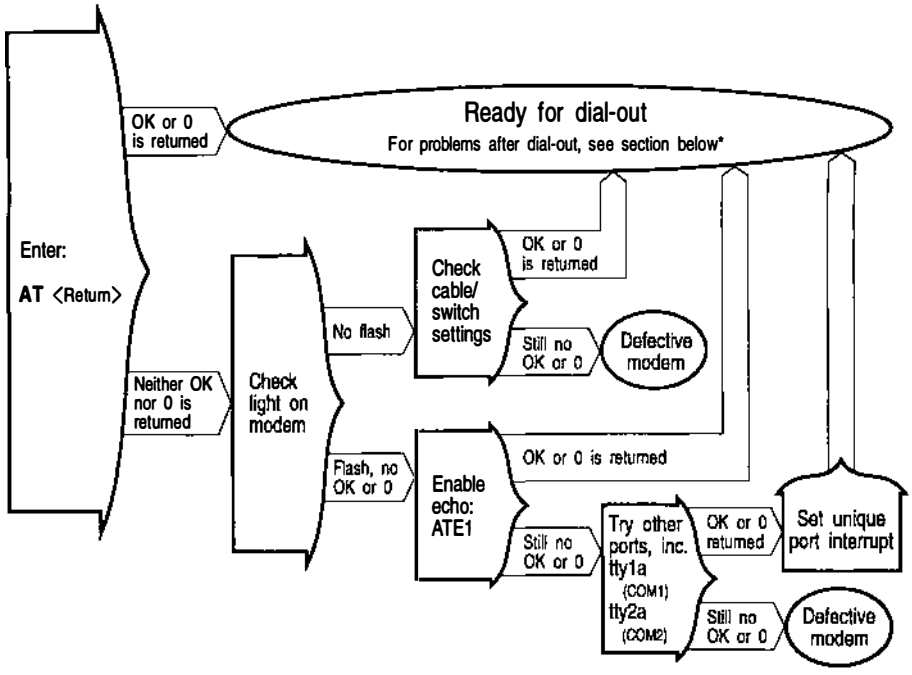


Figure 12-1 Testing your modem connection

11. If this does not occur, check that the "receive" light on the modem flashes when you press a key. This indicates the modem is receiving signals from the keyboard. If this light is not flashing, check your cable and modem switch settings. If the "receive" light flashes, but you still do not get an "OK" response from the modem, enable the modem's echo capability and responses to commands by entering:

    ATE1Q0

12. If your terminal still does not display "OK" or "0", connect the modem to another port (COM1 or COM2). If the modem works with the new port, check that the device is using the correct interrupt vector. (Serial port COM1 uses interrupt 4; COM2 uses interrupt 3.)

13. If the terminal does not echo the "OK" message when it is connected to the new port, then double check your cable. You can also connect a terminal to the port (with a proper terminal cable) and verify that the port works. If the modem returns garbage, then try connecting at different speeds; some modems can be set to communicate at the DTE connect speed, while others communicate at a fixed speed. If the modem still does not respond properly, the modem may be defective.

14. If you plan to use the modem primarily for dial-in, skip to step 15. To test the modem for dial-out, enter:

    ATDT*phonenumber* for tone dial phone connection

    ATDP*phonenumber* for pulse dial phone connection

    When you confirm that the modem can dial out, exit **cu** by entering:

    ~.

    Then press ⟨Return⟩.

    You are now ready to dial into another system. Use the following command to dial out:

    cu -x9 -ltty1A ***phonenumber***

    where ***phonenumber*** is number of the remote system. The **-x9** option generates useful debugging information.

    > **NOTE** When including a phone number in the **cu** command to connect with a modem, you should not use the comma (,) from the Hayes command set to indicate a pause. Use hyphens instead. This also applies to *Systems* file entries.

15. If the line is also to be used for dial-in, you must follow some additional steps to configure the modem. Some modems have switches or software commands for setting the modem configuration. If your modem has such settings, configure it as instructed in your modem manual. The modem should be configured as follows:

    - The modem must be set to Auto-answer. Your modem must support auto-answer; some modems do not have this setting.
    - It should not answer when the DTR (Data Terminal Ready) line is low, and should disconnect from the current connection when DTR goes from high to low.
    - The CD (Carrier Detect) line should follow the incoming carrier; it should go low when a carrier is present, high when a carrier is not present.
    - Echo should be turned off; it should not echo characters sent to it.
    - It should be set for quiet mode, sending no response codes.

16. Enable the port you are using for your modem with the **enable**(C) command:

    enable ***ttyname***

    where ***ttyname*** is the modem control port.

17. Dial this modem from another modem.

If you have any problems, refer to the section on troubleshooting modems in "Modem problems" (page 579).

Dialing into your computer

To allow dialing into your computer, you must enable a serial line that recognizes modem control signals with the **enable**(C) command.

To use the main serial adapter (COM1), enter:

```
disable tty1a
enable tty1A
```

For the alternate serial adapter (COM2), enter:

```
disable tty2a
enable tty2A
```

Note that tty1A and tty1a refer to the same (main) serial line, and tty2A and tty2a refer to the same (alternate) serial line. Do not enable the same line in both its modem and non-modem modes at the same time, as this causes an error.

Adding passwords for dial-in lines

If desired, you can define special dial-in passwords on selected tty lines, requiring selected classes of users to input dial-in passwords. Logging information, including the last time of connection, can be stored for later use.

Specific dial-in lines that require passwords are defined in the file */etc/dialups*. The format is one tty device name per line, for example:

```
/dev/tty1A
/dev/tty5C
```

The actual dialup passwords are kept in the file */etc/d_passwd*. The password format is the same one used in */etc/passwd*. The first field ("user name") in */etc/d_passwd* is not a user name, but the name of a shell program (for example, */bin/sh*) used in */etc/passwd*. If the login shell of the user attempting to log in (on a tty line listed in */etc/dialups*) is listed in */etc/d_passwd*, then the user is prompted for the dial-in password stored in */etc/d_passwd*.

Here is the syntax for creating a dial-in password:

passwd -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 super user can change a dialup shell password.

Dialing out from your computer

The **cu**(C) and **uucp**(C) utilities call remote systems and transfer data on UNIX systems. The file */usr/lib/uucp/Devices* (referred to as *Devices*) contains information used by these programs to determine the characteristics of a particular serial line.

The *Devices* file contains lines that specify the device for the line, the call-unit associated with the line, and the baud rate, which are to be used by UUCP. (Modem control devices should be used with lines connected to modems.)

Using dialer programs

For dialing, both **cu** and UUCP use a common set of dialers, which can be standalone binaries (programs) like */usr/lib/uucp/dialHA12*, or entries from the file */usr/lib/uucp/Dialers*.

Dialers file entries

Table 12-3 lists dialer types that are available as *Dialers* entries.

Table 12-3 Dialers file entries

Dialer type	Modem or data switch
direct	direct line; no dialer
penril	Penril modem
hayes1200	Hayes Smartmodem 1200 (or compatible)
hayes2400	Hayes Smartmodem 2400 (or compatible)
ventel	Ventel 212+ modem
vadic	Racal Vadic 3451 modem
vadic9600	Vadic 9600VP
develcon	Develcon network dataswitch
micom	Micom network dataswitch
rixon	Rixon Intelligent Modem
att4000	AT&T Programmable 300/1200 Modem Model 4000
att2212c	AT&T DATAPHONE II 2212C Modem
att2224	AT&T DATAPHONE II 2224 Modem
nls	Network Listener Service

The following is an example *Devices* entry that uses a *Dialers* file entry:

```
ACU tty1A - 2400 hayes2400
```

NOTE If you are experimenting with an unsupported modem, the "hayes2400" *Dialers* file entry is a good starting point; it can be used even with high-speed modems.

Although it is possible to create or modify a *Dialers* file yourself, the syntax can be difficult to follow. Consult the **dialers**(F) manual page for more information.

Dialer binaries

Table 12-4 lists the dialer binaries provided with your distribution. They are supplied in binary and source form.

Table 12-4 Dialer binaries

Binary file	Modem
dialHA12	Hayes Smartmodem 1200 or compatible
dialHA24	Hayes Smartmodem 2400 or compatible
dialHA96V	Hayes V-series Smartmodem 9600
dialMUL	Multitech MultiModem 224 EH or v.32
dialVA3450	Racal Vadic 3451 Modem
dialT1500	Trailblazer TB1500
dialTBIT	Telebit Trailblazer Modem
atdialHAY	Hayes Ultra 96 Modem 2000US
atdialMICC	MICC 9610 Modem
atdialMT	MULTITECH MT932EA Modem
atdialT25	Telebit T2500 Modem
atdialUSR	US Robotics Courier V.32bis Modem
atdialW96	US Robotics World Port 9600 Modem

There are two types of dialer binaries: *dial* and *atdialer*. The *dial* binaries (example *dialHA24*) are separate executable programs that must be recompiled from source files if you wish to customize them. The *atdialer* dialers (example: *atdialHAY*) are actually a single executable (*/usr/lib/uucp/atdialer*) with a configuration file that can be changed with any text editor. The sections that follow explain how to create both types of dialers.

Modem commands and setup strings

To create a new dialer, you need to understand how modem commands are used. You can enable or disable desired features by sending commands in a "setup string" to the modem. For example, the following setup string is used in the *dialHA24* dialer:

```
ATQ0E0T&D2&C1S0=0X4S2=043
```

You can change these strings to suit a different communications protocol or modem-specific commands by consulting the documentation for your modem. Though the setup commands may seem confusing because they are concatenated, there are two basic types of Hayes commands:

- modem commands (example: Q0, &D2)
- modem S-registers (example: S0=0)

Hayes-compatible command strings always begin with the AT (attention) command.

NOTE Consult your modem documentation for complete information on commands and options.

Creating a new dial dialer

You can customize a *dial* program by editing the source file (example *dialHA24.c*) and changing modem commands embedded in the C code and then recompiling the source.

The source and a makefile for recompiling *dial* binaries are included in the directory */usr/lib/uucp*. If you have any other kind of modem, you can modify any of the source files and create your own dialer program. Remember that you must have the Development System installed to compile a *dial* program.

To make a new *dial* program, follow these steps:

1. Change directory to */usr/lib/uucp*.
2. Copy the dialer program you wish to modify and make the desired changes. For example, the *dialHA24.c* setup commands are located in this code segment:

   ```
   #define MDSETUP          "ATQ0E0T&D2&C1S0=0X4S2=043\r"
   ```

   There are several "defines" in the program, each containing a different string, such as **MDHANGUP**, the command string to send that tells the modem to hang up (ATQ0H). Change each define as appropriate.

   NOTE The \r is not a modem command, but a string that indicates a carriage return; all modem commands in a dialer program require a carriage return as if they were being entered on the screen of your computer.

3. Edit the file *makefile* in the directory */usr/lib/uucp* and find the line that reads:

   ```
   EXES= dialHA12 dialHA24 dialHA96V dialMUL dialTBIT dialT1500 dialVA3450
   ```

   Add the name of the dialer program that you wish to use. When this is done, exit the file, saving the changes you made.

4. Next, enter the following command at your shell prompt:

   make

   When the **make** command is finished, you have a new dialer program. This can be used in the fifth field of an entry in the *Devices* field.

Creating a new atdialer dialer

You can create a new *atdial* dialer without having the Development System installed. An *atdial* dialer is actually a link to the binary */usr/lib/uucp/atdialer* that calls a configuration file in the */etc/default* directory. The configuration file contains all the commands specific to that modem. For example, *atdialHAY* is linked to *atdialer* and the configuration file is in */etc/default/atdialHAY*.

NOTE *atdial* dialers are for high speed modems. Many different modems are available in the marketplace. They range in speed from 300 to 38400 bps. SCO does not support serial communication speeds greater than 9600 bps. In some cases, higher speeds are known to work. Some modems also have data compression while others do not.

Table 12-5 atdialer configuration strings

String	Meaning	Example
MDM_SETUP	Setup string	AT&FE0TL0&D2&C1&Q5
MDM_OPTION	Special option*	
MDM_DIALCMD	Dial command	ATD
MDM_ESCAPE	Escape to command mode	+++
MDM_HANGUP	Hangup	ATQ0H0
MDM_RESET	Reset	ATQ0Z0
MDM_DIALIN	Setup for dial-in	ATS0=1Q1
MDM_ATTN	Attention	AT
MDM_DSBLESC	Disable modem escape	ATS2=128
RTC_*speed*	Result codes†	CONNECT 9600

* This feature allows you to set a special register when you put an X or an x at the end of the phone number in a *Systems* file entry. For example, you can send **ATSP** to do UUCP spoofing on a phone number with an "x" at the end. This feature is only valid when used in a *Systems* file entry; it cannot be used with a telephone number on the **cu** command line.

† Configuring the connect speed strings is an important aspect of creating a dialer. See "Editing the connect speed strings" (page 319) for more information.

Modems that vary line speed

Some high speed modems can run the serial port at specified speeds, while the modem varies the line speed as needed. The modem must be prepared for this option by putting the appropriate commands in the **MDM_OPTION** entry then modifying */usr/lib/uucp/Devices* and */etc/inittab* to use the highest available

speed. You must also edit the configuration file so that the strings containing the connect speed for all connections read "not used", except for the 9600 entry. See the next section for more information.

Editing the connect speed strings

The **RTC_*speed*** strings in the */etc/default/atdial** configuration files allow the dialer to recognize connect messages from the modem and map them to appropriate speed(s). For example:

```
RTC_9600=CONNECT 9600
```

The dialer then recognizes `CONNECT 9600` as a 9600 bps connection. The connect messages for each speed must be unique or the first speed mapped to the connect message is used. For unused speeds, simply set the **RTC_*speed*** string to *not used*. For example:

```
RTC_300=not used
```

Normally, all but one **RTC_*speed*** string is set to *not used*. This is because most 9600 bps modems use "speed conversion". They communicate with the host serial port at 9600 bps even if they connect to the remote modem at a slower speed. The **RTC_*speed*** strings for these modems generally look like this:

```
RTC_9600=CONNECT
```

Any modem result message that begins with `CONNECT`, such as `CONNECT 9600` or even `CONNECT 2400`, is recognized as a 9600 bps connection and accepted. It is possible to refine this by specifying all valid connect messages. For example:

```
RTC_2400=CONNECT 2400
RTC_9600=CONNECT 9600
```

For a modem that performs speed conversion, this more accurate setup is only necessary if you need to guard against long UUCP transfers at 2400 bps. This is because the speed recognized by the dialer is matched against the speed range in the *Systems* file or the speed range on the **cu** command line. If it is out of range, an `EXECDIAL LOCAL FAILURE` will result. In the following command the speed range specified is 9600-9600:

cu -ltty1A -s9600 5551212

Connections at 2400 bps using the above command will fail if separate **RTC_*speed*** lines are set up. The following commands allow a 2400 bps connection in this case:

cu -ltty1A -s2400 5551212

cu -ltty1A -s2400-9600 5551212

RTS/CTS flow control

Flow control is used between the computer and the modem to start and stop the flow of data when necessary (as when the modem's input buffer is full, or the computer needs to perform a disk write). There are two types of flow control: hardware (RTS/CTS) and software (XON/XOFF). RTS/CTS is the preferred and most efficient method.

The modem must also be configured for hardware flow control. For example, the *atdialHAY* dialer supports RTS/CTS flow control with the **&K3** command.

RTS/CTS on outgoing calls

RTS/CTS flow control for outgoing calls is supported by the *atdialer*, *dialHA96V*, *dialMUL*, and *dialTBIT* binary dialers. All of these dialers recognize strings of the form **STTY=*settings*** in their */etc/default/**dialer*** files. For example, you can enable bidirectional hardware flow control for the *dialTBIT* dialer by creating a file */etc/default/dialTBIT* that contains the following line:

```
STTY= RTSFLOW CTSFLOW
```

RTS/CTS on incoming calls

RTS/CTS flow control for incoming calls is configured by adding the **RTSFLOW CTSFLOW** flags to the appropriate */etc/gettydefs* entry specified in the */etc/inittab* entry for the modem tty port.

The **CTSFLOW** flag is often used by itself to prevent the modem buffer from overflowing when speed conversion is used for incoming calls. For example, assume a MultiModem v.32 is set for a fixed DTE speed of 9600 bps (the default). When a 2400 bps calls in, then the computer will send data to the modem at 9600 bps, which the modem can only send out at 2400 bps. The **CTSFLOW** flag in */etc/gettydefs* tells the serial driver to stop sending data when the modem drops CTS. This allows the modem to signal when it is busy and prevent data overrruns.

atdialer creation procedure

To create a new *atdialer*, do the following:

1. Log in as *root*.
2. Change directories to */usr/lib/uucp*.
3. Enter the following command:

   ln atdialer atdial*new*

   where ***new*** is the name of the dialer you wish to create.
4. Change directories to */etc/default*.

5. Copy an existing configuration file that you can modify. For example:

   cp atdialHAY atdial*new*

   Be sure and use the same name you used in step 3.

6. Edit the configuration file as desired. Consult Table 12-4 for a list of file entries.

Chapter 13

Building a remote network with UUCP

This chapter explains how to use UUCP to build a remote network system for your computer using a normal telephone line and a modem.

NOTE UUCP is not a terminal emulation program. If you want to use your modem to dial into another computer and log on, you should refer to the "Using modems" chapter of this guide and follow the instructions for adding dial-in and dial-out modems.

If you plan to do extensive file transfers between physically separated XENIX and UNIX systems, you should set up a UUCP connection.

What is UUCP?

The UUCP package permits XENIX and UNIX systems to communicate as part of a remote network. The name UUCP is an acronym for "UNIX-to-UNIX Copy". The UUCP package consists of a group of programs that provide the following capabilities:

- remote file transfer (**uucp**)
- remote command execution (**uux**)
- mail to and from remote sites (via **mail**)

The UNIX system uses the HoneyDanBer implementation of UUCP. Used primarily over phone lines, UUCP can connect with specific remote machines on a demand or scheduled basis, and by either dialing out or allowing other machines to call in.

UUCP uses a batch method to manage communications traffic, storing (or "spooling") requests for later execution when actual contact is made between systems. When UUCP commands are executed, work files and any data files needed are created in */usr/spool/uucp* and its subdirectories. 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).

How to use this chapter

This chapter describes how to build a UUCP system and covers both hardware installation and software configuration. There are also sections on routine maintenance and troubleshooting.

The following is a procedural outline of what must be done to set up your UUCP network:

1. Connect and configure a modem or direct wire.
2. Configure the UUCP software by editing the configuration files, or by using the **uuinstall** utility.
3. Create login accounts for any sites that will be calling your system.
4. Test your connections with each remote site.

NOTE If you are planning to route mail over your UUCP system, see the chapter "Setting up electronic mail" in this guide for instructions on configuring mail traffic to work over UUCP.

The most important task of configuring UUCP is the editing of several control files that act as the database for UUCP. The next few sections describe the function of these files, and "Configuring UUCP on your system" (page 330) explains the information that these files contain. You can edit these files manually, or you can use **uuinstall**, a utility that edits them for you and explains each entry. **uuinstall** also includes an extensive help facility. Read "Configuring UUCP on your system" (page 330) carefully before running **uuinstall** so that you understand the UUCP database.

What you need

To set up your UUCP communication system, you need:

- at least one RS-232 serial line (or serial port) on your computer to use for UUCP
- the UUCP and MAIL packages extracted from your UNIX system distribution using **custom**(ADM)
- a modem, configured as described in the "Using modems" chapter of this guide
- a standard telephone jack for access to the telephone system
- a cable to connect the serial port to the modem

UUCP commands

UUCP programs are divided into two categories: user programs and administrative programs. The paragraphs that follow describe the programs in each category.

User programs

The user programs for basic networking are in */usr/bin*. No special permission is needed to use these programs, although it is possible to restrict access to the devices they control. These commands are all described in the "Communicating with other sites" chapter of the *User's Guide*.

cu allows you to call out to a remote computer so you can be logged in on both at the same time. You can transfer files or execute commands on either computer without dropping the initial link.

ct connects your computer to a remote terminal so the user of the remote terminal can log in. The user of a remote terminal can call the computer and request that the computer call it back. The computer then drops the initial link so that the remote terminal's modem is available when it is called back.

uucp copies files from one computer to another. It creates work files and data files, queues the job for transfer, and calls the **uucico** daemon, which contacts the remote computer.

uupick retrieves the files placed in */usr/spool/uucppublic/receive* when files are transferred using **uuto**. It also provides a means of controlling queued transfers.

uuto copies files from one computer to a public spool directory on another computer in */usr/spool/uucppublic/receive*. Unlike **uucp**, which lets you copy a file to any accessible directory on the remote computer, **uuto** places the file in an appropriate spool directory, and tells the remote user to pick it up with **uupick**.

uux creates the work, data, and execute files needed to execute commands on a remote computer. The work file contains the same information as work files created by **uucp** and **uuto**. The execute files contain the command string to be executed on the remote computer and a list of the data files. The data files are those files required for the command execution.

Administrative programs

Most of the administrative programs, control files, and scripts are in */usr/lib/uucp*. Two exceptions are **uuinstall** and **uulog**, which are in */etc* and */usr/bin*, respectively.

uucheck checks for the presence of basic networking directories, programs, and support files. It also checks the *Permissions*, *Systems*, and *Devices* files for syntax errors.

uuclean cleans up the spool directory. It is normally executed from a shell script called **uudemon.clean**, which can be set up to be run by **cron**.

uulog displays the contents of a specified computer's log files. Log files are created for each remote computer your computer communicates with. The log files contain records of each use of **uucp**, **uuto**, and **uux**.

uustat displays the status of requested transfers (**uucp**, **uuto**, or **uux**).

uutry tests call-processing capabilities and displays debugging information. It invokes the **uucico** daemon to establish the communications link.

UUCP directories

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 publically 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:

Systems contains information needed to establish a link to a remote computer, including the name of the connecting device associated with the remote computer, when the computer can be reached, telephone number, login sequence, and password.

Permissions defines the access level granted to computers when they attempt to transfer files or remotely execute commands on your computer.

Devices contains information concerning the port name, speed, and type of the Automatic Call Units (modems), direct links, and network devices.

UUCP background programs

The **uucp** traffic is managed by three *daemons*, or supervisory programs, that run in the background, handling file transfers and command executions. (The daemons can also be executed manually as commands.)

uucico selects the device used for the link, establishes the link to the remote computer, performs the required login sequence and permission checks, transfers data and executes files, logs results, and (if requested) notifies the user by mail of transfer completions. When the local **uucico** daemon calls a remote computer, it "talks" to the **uucico** daemon on the remote computer during the session.

uuxqt 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.

How UUCP works

When you enter a UUCP command, the program creates a work file and usually a data file for the requested transfer. The work file contains information required for transferring the file(s). The data file is a copy of the specified source file. After these files are created in the spool directory, the **uucico** daemon is started.

The **uucico** daemon attempts to establish a connection to the remote computer. First it gathers the information required for establishing a link to the remote computer from the *Systems* file. This is how **uucico** knows what type of device to use in establishing the link. Next, **uucico** searches the *Devices* file looking for the devices that match the requirements listed in the *Systems* file. After **uucico** finds an available device, it attempts to establish the link and log in on the remote computer.

When **uucico** logs in on the remote computer, the **uucico** daemon is started on the remote computer. The two **uucico** daemons then negotiate the line protocol to be used in the file transfer(s). The local **uucico** daemon then transfers the file(s) that you are sending to the remote computer. The remote **uucico** places the file in the specified pathname(s) on the remote computer. After your local computer completes the transfer(s), the remote computer may send files that are queued for your local computer. The remote computer can be denied permission to transfer these files with an entry in the *Permissions* file. (This is also affected by directory permissions.) If this is done, the remote computer must establish a link to your local computer to perform the transfers. A remote computer can also request files.

If the remote computer or the device selected to make the connection to the remote computer is unavailable, the request remains queued in the spool directory. If set up to run by **cron** each hour, **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 is created in the */usr/spool/uucp/obie* directory, *C.obieNxxxx*, 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.

Table 13-1 Example UUCP control files (sites: kilgore and obie)

Systems:
```
obie Any ACU 2400 14081234567 --ogin:-BREAK-ogin:
 nuucp ssword: mavra
```

Devices:
```
ACU tty1A - 2400 dialHA24
```

Permissions:
```
LOGNAME= ukilgore MACHINE= kilgore \
  READ=/usr/spool/uucppublic:/usr/kilgore \
  WRITE=/usr/spool/uucppublic:/usr/kilgore \
  REQUEST=no  SENDFILES=call \
  COMMANDS=rmail:rnews:uucp
```

7. **uucico** creates a lock file (*LCK..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 (via **uux**), an execute *X.file* is created in the */usr/spool/uucp* directory. The **uuxqt** daemon scans this directory for work, checks the *Permissions* file to confirm permission to execute the command, then executes it. This takes place after the modems hang up and **uucico** exits.

Configuring UUCP on your system

To configure your UUCP system, you must connect a modem and edit a series of files that contain information about, and control the actions of, the UUCP programs.

NOTE To configure UUCP over a direct wire, see "Connecting two local systems using a direct wire" (page 360) for instructions.

The UUCP control files are in the */usr/lib/uucp* directory. You can modify these files with a standard text editor, or you can use the **uuinstall** utility. The descriptions found in "Detailed descriptions of UUCP configuration files" (page 334) provide details on the structure of these files so that you can create more complex configurations than the examples provided.

NOTE After configuring UUCP, if you have any problems initiating transactions, see "UUCP troubleshooting" (page 604) for helpful information.

Connecting a UUCP modem

To configure and install a modem, follow the instructions in the "Using modems" chapter of this guide and return to this section after your modem is up and running.

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 highst commom 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.

Editing the UUCP control files

This section is concerned with the configuration or control files that act as the UUCP database. A simple configuration is assumed in this section; more detailed descriptions of the UUCP files are found later in this chapter.

To configure the UUCP files, do the following:

1. Set up the *Systems* file on each machine. Use the following format:

   sitename `Any ACU` ***baud phone#*** `-\r\d-ogin:-\K\d-ogin:-\K\d-ogin:-\K\d-ogin: nuucp word:` ***password***

   where:

sitename	login name for the opposite site
baud	baud rate (or range) used for dial-out
phone#	phone number of opposite site
password	password for UUCP account at opposite site

   NOTE Some lines may exceed 80 characters in length, but should still be treated as a single line. In other words, do not press ⟨Return⟩ when the text reaches the right side of the screen. Press ⟨Return⟩ only when you have finished typing in the line.

   Sitenames should be no longer than seven characters and should contain no 8-bit characters (some sites reject both). In addition, they should not contain control characters or escape sequences.

   The baud rate should be set to the highest common baud rate between the modems that will be used. In other words, if the modem on machine A is a 2400 baud modem, and the modem on machine B is a 1200 baud modem, then these should both be set for 1200.

   Here is a set of example entries for a pair of sites, *kilgore* and *obie*, with 2400 baud modems:

   site: obie

   ```
   kilgore Any ACU 2400 5551212 -\r\d-ogin:-\K\d-ogin:-\K\d-ogin:-\K\d-ogin:
    nuucp word: TrouTster
   ```

   site: kilgore

   ```
   obie Any ACU 2400 5551212 -\r\d-ogin:-\K\d-ogin:-\K\d-ogin:-\K\d-ogin:
    nuucp word: mAvraC
   ```

NOTE If you are configuring UUCP to function over a leased line, use "Direct" in place of "ACU") in the *Systems* entries. The same is true for direct wire connections.

There are many other specifications that can be included in the *Systems* file; it is discussed in more detail in "Adding entries for remote sites to the Systems file" (page 334).

2. Set up the *Permissions* file on each machine. Add the following text to the bottom of the file */usr/lib/uucp/Permissions* on each machine, leaving a blank line between any text already in the file and the following new text, substituting the respective site name for ***site***:

```
MACHINE=site LOGNAME=login \
COMMANDS=rmail:rnews:uucp \
READ=/usr/spool/uucppublic:/usr/tmp \
WRITE=/usr/spool/uucppublic:/usr/tmp \
SENDFILES=yes REQUEST=yes
```

Note that the **LOGNAME** must be the login name that the site uses. There are many other options that can be included in the *Permissions* file; it is discussed in more detail in "Limiting access with the Permissions file" (page 341).

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 (see the "Administering user accounts" chapter in this guide), 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, follow these steps:

1. Choose a new user name and a user ID (identification number) for the UUCP login. The name can be any combination of letters and digits that is no more than eight characters long. The user ID must be an integer in the range 50 to 65535.

   Make sure the name and ID are unique. A UUCP login entry must not have the same name or ID as any other login entry.

2. To create the new account, invoke **sysadmsh** and make the following selection:

   Accounts ➪ User ➪ Create

3. Use the following information to create the account:

   Login shell: */usr/lib/uucp/uucico*
   Home directory: */usr/spool/uucplogins/username*

   Passwords are optional, but recommended, for UUCP logins.

UUCP anonymous login accounts

UUCP login accounts are created with a default password expiration of 14 days. To alter this, you must use the **sysadmsh**(ADM) selection Accounts ➪ User ➪ Examine:Expiration to redefine this limit. For more information, see the "Administering user accounts" chapter in this guide.

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 extend the number of login attempts by selecting Accounts ➪ User ➪ Examine:Logins . If the account was locked due to too many unsuccessful login attempts, the "Account Locked" field displays "Too many unsuccessful login attempts". You can clear this condition either by setting the maximum unsuccessful logins to a larger number (including infinite), or by selecting the Lock status option Clear all locks.

Testing the UUCP connection

To test your UUCP connection, follow these steps:

1. If you are configuring an outbound UUCP connection, **cu** must function in order for UUCP to work. Be sure and follow the instructions for configuring and testing the modem in the "Using modems" chapter of this guide.
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.

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*

Detailed descriptions of UUCP configuration files

This section includes detailed information on the UUCP database files. Sites with specific needs can design files as needed rather than relying on the simple examples included in this chapter.

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 the "Setting up electronic mail" chapter of this 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 "Using Dialcodes to create a portable Systems file" (page 353) .
login-script	contains login information (also known as a "chat script").

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 (**Mo Tu We Th Fr**)
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 340) 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 discussion of *Permissions* file).

The optional ***time*** subfield should be a range of times in 24-hour clock format, such as 0800-1230. If no ***time*** is specified, any time of day is assumed to be allowed for the call. A time range that spans 0000 is permitted. For example, 0800-0600 means all times are allowed other than times between 6 AM and 8 AM.

For example, the following permits calls on Mondays, Wednesdays, and Fridays between the hours of 9 am and noon (the ***schedule*** field is in boldface for clarity):

```
grebe MoWeFr0900-1200 ACU D1200 14087672676 ogin:
 nuucp ssword: Crested
```

You can also specify more than one set of ***day*** and ***time*** entries by separating them with commas. This is useful for more complex specifications. The following example allows calls from 5 PM to 8 AM, 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 340) 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 documention 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 discussion on the *Devices* file, ***speed*** field). Some devices can be used at any speed, so the keyword **Any** can be used. However, we recommend that you specify the actual range of speeds that can be used. (If **Any** is used in both *Systems* and *Devices* entries, 1200 is assumed.) For example, this field must 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.

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 "Using Dialcodes to create a portable Systems file" (page 353) 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 is given as a series of space-separated fields and subfields of the following format:

expect send

where ***expect*** is the string that is received, and ***send*** is the string that is sent when the ***expect*** string is received.

The ***expect*** field can be made up of subfields of the following form:

***expect*[*-subsend-subexpect*] ...**

where the ***subsend*** is sent if the prior ***expect*** is not successfully read and the ***subexpect*** following the ***subsend*** is the next expected string. To make this distinction clear: the send-expect sequence sends a string if the expect string is received; the subsend-subexpect sends only if the prior expect string is not received within 10 seconds.

For example, with "login:–login:", the UUCP program expects "login:". If a "login:" is received, it goes on to the next field. If it does not get "login:", it sends nothing followed by a carriage return, then looks for "login:" again. If no characters are initially expected from the remote computer, the null string ("") should be used in the first ***expect*** field. Note that all ***send*** fields are sent followed by a carriage return unless the ***send*** string is terminated with a " \c ".

If an ***expect*** string starts with a dash, it is interpreted as a null ***expect*** string followed by a ***subsend*** string. For example, "--login:" sends a carriage return and then expects a "login:".

The ***expect*** string need not be complete; only the trailing characters must be specified, as in "ogin:". This avoids difficulties with login strings that use an uppercase letter as in "Login:" or "Password:", and also difficulties when the line is shared by dial-in and dial-out.

Creating login scripts

This section explains in greater detail how to create a login (chat) script.

Consider the following sample *Systems* file entry:

```
terps Any ACU 1200 18005211980 "" \r ogin:-BREAK-ogin:
 uucpx word: ichore
```

This is how this script would work during connection:

1. Nothing is expected initially.
2. A carriage return is sent and the script waits for the prompt "ogin:" (login:).
3. If it does not receive "ogin:", send a **BREAK** signal.
4. When "ogin:" is finally received, send the login name *uucpx*.
5. When the prompt "word:" (for Password:) is received, send the password *ichore*.

Login (chat) scripts often require some experimentation. There are cases that require one or more **BREAK** sequences before presenting a login (this is often true with variable speed modems). If you cannot obtain the necessary login sequence from the system administrator for a given site, it is a good idea to connect with the site manually. You can accomplish this using **cu** and find out what must be sent to generate a login prompt. (You can also connect with a system using a **uutry** for debugging; see "Debugging UUCP communications" (page 609) for details. There are several escape characters that cause specific actions when sent during the login sequence, some of which correspond to keystrokes; these should be included in the script where necessary. See Table 13-2.

Table 13-2 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.

(Continued on next page)

Table 13-2 Login (Chat) script escape sequences
(Continued)

Character	Description
\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.

Setting up polling

Use **uudemon.poll2** to set up polling. 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 * * *       uudemon.poll2 -d
0 * * * *       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, you 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 where Sunday is 0.

Limiting access with the Permissions file

If other machines will be dialing into your system, the *Permissions* file (*/usr/lib/uucp/Permissions*) specifies the permissions that remote computers have with respect to login, file access, and command execution. There are options that restrict the remote computer's ability to request files and its ability to receive files queued by the local site. Other options specify the commands that a remote site can execute on the local computer.

Structuring Permissions file entries

Each entry is a logical line with physical lines terminated by a " \ " to indicate continuation. Entries are made up of options delimited by spaces. Each option is a name-value pair in the following format:

name=value

NOTE 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 they are used, and lists their default values.

REQUEST

specifies whether the remote computer can request to set up file transfers from your computer. When a remote computer requests to receive a file, this request can be granted or denied. The following string specifies that the remote computer can request to transfer files from your computer:

```
REQUEST=yes
```

The following string specifies that the remote computer cannot request to receive files from your computer:

```
REQUEST=no
```

The no value 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 appropraite 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 with 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. Likewise, */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. So, 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
```

1. The **ALL** value can appear anywhere in the string; and 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**.
2. 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. Therefore, the real question is 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=/
```

NOTE The following command provides a useful interpretation of the *Permissions* file:

/usr/lib/uucp/uucheck -v | more

Adding dial-out entries to the Devices file

The *Devices* file (*/usr/lib/uucp/Devices*) contains information for all the devices that can be used to establish a link to a remote computer. Devices are Automatic Call Units, direct links, or network connections. This file works closely with the *Dialers*, *Systems*, and *Dialcodes* files. Before you make changes in any of these files, you should be familiar with them all. A change to an entry in one file may require a change to a related entry in another file.

Each entry in the *Devices* file has the following format:

type ttyline dialerline speed dialer-token

where:

type 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**. For more information about configuring UUCP over TCP/IP, see "Configuring UUCP over TCP/IP" (page 354).

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 ttyla - 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 352).

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 1200 bps. If a device can be used at a range of speeds, then the speed field can specify this range (for example, 1200-9600 or D1200-9600). This is preferable to the use of **Any**.

The dialer-token field

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 "Dialing out from your computer" (page 315).

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 352) .) 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
  ```

  As shown, the ***token*** portion is "\D", which indicates that it is retrieved from the *Systems* file without translation. The *Systems* file entry for this particular computer contains the token in the ***phone*** field; this is normally reserved for the phone number of the computer (refer to *Systems* file, ***phone*** field). The "\D" ensures that the contents of the ***phone*** field is not interpreted as a valid entry in the *Dialcodes* file.

- Modems used with a local network switch

  If an automatic dialing modem is connected to a switch, your computer must first access the switch and the switch 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.

The following are two escape characters that can appear in the *dialer-token* field:

\T indicates that the ***Phone*** field should be translated at this stage, using the *Dialcodes* file. This escape character is normally placed in the *Dialers* file for each caller script associated with an automatic dial modem (penril, ventel, and so on). The translation 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).

Special UUCP configuration options

This section contains several options that are used for 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**). Table 13-3 lists the available protocols.

Table 13-3 UUCP communications protocols

Protocol	Description
g	standard UUCP protocol for connection over serial lines and modems. Uses error correction.
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).

Adding dialers to the Dialers file

The *Dialers* file (*/usr/lib/uucp/Dialers*) specifies the initial conversation that must take place on a line before it can be made available for transferring data. New entries can be added to this file if your modem does not appear in the file. See "Using dialer programs" (page 315) for more information.

Using Dialcodes to create a portable 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.

Creating alternate control files with Sysfiles

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.

Changing uucico packet parameters

An added feature is the ability to change two specialized parameters contained in the **uucico** program without having to recompile the source. These parameters are:

windows — specifies the size of window that the sliding-window protocol should use.

pktime — is the number of seconds **uucico** should wait before giving up and re-transmitting the packet being sent.

See the **uucico**(ADM) manual page for more information.

NOTE Changing the **uucico** windows parameters can cause compatibility problems. For example, it is best not to modify the default values if you plan on connecting to UUNET.

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 over TCP/IP

There are several reasons you may wish to configure UUCP over TCP/IP. Some sites do not provide **ftpd** or **rshd** servers, and/or their respective clients, **ftp** and **rcp**. Thus transferring files across a TCP/IP network is not an option with these systems. Additionally, some versions of TCP/IP do not provide the *S*imple *M*ail *T*ransfer *P*rotocol, SMTP, for mail transfer. In both of these cases, setting up UUCP to use TCP/IP may be an option to allow both file transfer and exchange of mail between such systems.

In the following paragraphs, the word "client" refers to a system which executes the **uucp** or **uux** command, while the term "server" or "listener" refers to a system that responds to a request from a client system.

There are two approaches to configuring UUCP over TCP/IP:

TCP socket interface
: The server system uses the **inetd** superserver to listen for incoming **uucp** requests on TCP port 540. When receiving a request from a client **uucico** process, the server system forks the **uucpd** daemon, which logs in the client with the shell **uucico**. The two **uucico** processes can now transfer information similar to the way they would in a standard serial line configuration.

TLI (Transport Layer Interface)
: The server system uses a process called **listen** to wait for requests from a predefined TCP port and then when receiving a request, forks a **uucico** process directly, bypassing the standard UUCP login sequence.

NOTE When connecting two SCO systems, the TCP socket interface is the preferred method. When connecting an SCO system to another vendor's TCP/IP product, consult the vendor's documentation to determine which method is supported and what procedure to use at that end. If the socket interface is supported, use it for best results.

Configuring UUCP over TCP/IP with the TCP socket interface

The following steps demonstrate how to configure UUCP over TCP/IP using the TCP socket interface between the systems *colossus* and *guardian*.

NOTE Before proceeding, verify that TCP/IP is installed and running properly.

1. On both systems, verify that the file */etc/inetd.conf* has the following line, and that it is uncommented (with no "#" character at the beginning of the line):

   ```
   uucp    stream tcp      nowait  NOLUID  /etc/uucpd uucpd
   ```

   NOTE If your system is running SCO TCP/IP Release 1.1.3, replace "NOLUID" with "root" in the above line.

   Should the program */etc/uucpd* not exist, execute the following commands as *root*:

   ln /usr/lib/uucp/uucpd /etc/uucpd
   chmod 755 /etc/uucpd

2. Verify that both systems have the following line in the */etc/services* file:

   ```
   uucp    540/tcp         uucpd           # uucp daemon
   ```

3. Verify the configuration of the *nuucp* account on each machine. Make sure that the shell of this account is **/usr/lib/uucp/uucico**, and that the account has a password on both machines. You will probably have to set a password for the *nuucp* account. Use the **sysadmsh** selection Accounts ⇨ User ⇨ Examine to confirm the account information and set a password.
4. Add the following line to the file */usr/lib/uucp/Systems* on *colossus*:

   ```
   guardian Any TCP,e Any - ogin: nuucp word: password
   ```

   Where ***password*** is the password of the *nuucp* account on *guardian*.
5. Add the following line to the file */usr/lib/uucp/Systems* on *guardian*:

   ```
   colossus Any TCP,e Any - ogin: nuucp word: password
   ```

   Where ***password*** is the password of the *nuucp* account on *colossus*.
6. Verify that the */usr/lib/uucp/Permissions* file on *colossus* has the following entry:

   ```
   MACHINE=guardian LOGNAME=nuucp \
           COMMANDS=rmail:rnews:uucp \
           READ=/usr/spool/uucppublic:/usr/tmp \
           WRITE=/usr/spool/uucppublic:/usr/tmp \
           SENDFILES=yes REQUEST=yes
   ```

7. Verify that the */usr/lib/uucp/Permissions* file on *guardian* has the following entry:

   ```
   MACHINE=colossus LOGNAME=nuucp \
           COMMANDS=rmail:rnews:uucp \
           READ=/usr/spool/uucppublic:/usr/tmp \
           WRITE=/usr/spool/uucppublic:/usr/tmp \
           SENDFILES=yes REQUEST=yes
   ```

8. Add the following line to */usr/lib/uucp/Devices* on both machines:

   ```
   TCP TCP,e       -       Any     TCP     540
   ```

9. Verify that both systems are in their partner's */etc/hosts* file, or resolvable by a nameserver.
10. If you needed to change anything in step 1, shutdown and reboot the systems where you made the changes. UUCP should then work as expected between the two systems. Each system should be able to issue requests for the opposite system.

NOTE Note that steps 3 through 7 are similar to the steps used in configuring UUCP between systems over a serial line. Also, the user names and sample *Permissions* files shown here are only examples and may be changed to suit the security needs of a particular site.

Configuring UUCP over TCP/IP with TLI

The following steps demonstrate how to configure UUCP over TCP/IP using TLI between the systems *colossus* and *guardian*. The example is a configuration where the machine *colossus* initiates all UUCP requests, while the machine *guardian* listens and responds to all requests. If you want both machines to be able to use the **uucp** command to initiate a UUCP transaction, repeat this procedure twice, exchanging machine names where appropriate.

NOTE Before proceeding, verify that TCP/IP is installed and running properly.

1. Pick a TCP port on which *guardian* will listen for incoming UUCP requests. The following example uses port 698. The port chosen should not already appear in the */etc/services* file, and should not be port 540.
2. Determine the address on which *guardian* will listen for requests. This number is the representation of a **sockaddr_in** structure as defined in the header file */usr/include/sys/netinet/in.h*. This number is a 16 byte number and needs to be constructed both in hexadecimal and in octal. Use of the **bc**(C) command is helpful here. To construct the number in hexadecimal, take the following steps:
   A. The first byte represents the address family. This is always **AF_INET**, or **02**, as defined in the header file */usr/include/sys/socket.h*.
   B. The second byte is always 00.
   C. The third and fourth bytes are the hexadecimal representation of the port on which *guardian* listens for UUCP requests. In our case, decimal 698 equals **2ba** in hexadecimal, hence the third byte is **02**, and the fourth byte is **ba**.
   D. The fifth, sixth, seventh, and eighth bytes represent the hexadecimal equivalent of each octet of the IP number of *guardian*. If the IP number of *guardian* is 132.147.144.77, then the fifth, sixth, seventh, and eighth bytes are **84**, **93**, **90**, and **4d** respectively.
   E. The last eight bytes are all **00**.
   F. Putting it all together, the address constructed is:

      ```
      02|00|02|ba|84|93|90|4d|00|00|00|00|00|00|00|00
      ```

      for a total of 16 bytes. The bars (|) are just for clarity.
   G. To create the octal representation of the address, convert each hexadecimal byte above into octal:

      ```
      \002\000\002\272\204\223\220\115\000\000\000\000\000\000\000\000
      ```

3. Set up the Network Listening Service as follows on *guardian*:

   A. As *root*, execute:

   nlsadmin -i inet/tcp

   This creates the directory */usr/net/nls/inet/tcp*, and allows us to add services that listen on the device */dev/inet/tcp*.

   B. Execute:

   nlsadmin -a 101 -c "/usr/lib/uucp/uucico -r0 -iTLI -unuucp"
   -w uucp -y "TLI UUCP" inet/tcp

   This command should be all on one line. (Do not use a backslash.) This adds service number 101, which executes the command **/usr/lib/uucp/uucico -r0 -iTLI -unuucp** as the user *uucp*, when a request for service number 101 is received on the device */dev/inet/tcp*.

   The options and arguments to **nlsadmin** are:

-a 101	Add a service with label 101. (Service code)
-c "*command*"	Fork the ***command*** in quotes
-w uucp	Execute the command given in the **-c** option as the user *uucp*
-y "TLI UUCP"	Comment
inet/tcp	This service is to listen on */dev/inet/tcp*. This step requires that the command in step 3A has already been executed.

   The options to the **uucico** command in the **-c** option of **nlsadmin** are:

-r0	Act in slave mode
-iTLI	Use the TLI interface
-u nuucp	Start up as the user *nuucp*. This is different from the **-w** option in the **nlsadmin** command.

   C. Execute the command:

   nlsadmin -l "\x020002ba8493904d0000000000000000" inet/tcp

   This is the hexadecimal representation of the address constructed in step 2.

4. Add the following line to */usr/lib/uucp/Systems* on *guardian*:

```
colossus Never TLI,e Any
```

   If this line for *colossus* has already been added for TLI so that *guardian* can initiate a connection, make no changes to the */usr/lib/uucp/Systems* file.

5. Add the following set of lines to */usr/lib/uucp/Permissions* on *guardian*:

```
MACHINE=colossus LOGNAME=nuucp \
        COMMANDS=rmail:rnews:uucp \
        READ=/usr/spool/uucppublic:/usr/tmp \
        WRITE=/usr/spool/uucppublic:/usr/tmp \
        SENDFILES=yes REQUEST=yes
```

6. Add the following line to the */usr/lib/uucp/Systems* file on *colossus*:

```
guardian Any TLI,e Any address
```

   where ***address*** is the octal representation of the address constructed in step 2. Thus the line to add is:

```
guardian Any TLI,e Any \002\000\002\272\204\223\220\115
        \000\000\000\000\000\000\000\000
```

   This should all be on one line. The backslashes (\) surrounding each octal byte are required.

7. Add the following line to the */usr/lib/uucp/Devices* file on *colossus*:

```
TLI inet/tcp - Any TLI \D nls
```

8. Add the following line to the */usr/lib/uucp/Dialers* file on *colossus*:

```
nls "" "" NLPS:000:001:101:\N\c
```

   Note that the "101" specified here matches the service code added with **nlsadmin -a** in step 3B.

9. Add the following lines to the */usr/lib/uucp/Permissions* file on *colossus*:

```
MACHINE=guardian LOGNAME=nuucp \
        COMMANDS=rmail:rnews:uucp \
        READ=/usr/spool/uucppublic:/usr/tmp \
        WRITE=/usr/spool/uucppublic:/usr/tmp \
        SENDFILES=yes REQUEST=yes
```

10. Verify that each machine is listed in the other's */etc/hosts* file or resolvable by the nameserver. Add the following lines to the end of the file */etc/rc2.d/S87USRDAEMON* on *guardian*, so that the listener is started when the system is booted:

```
nlsadmin -s inet/tcp 2>/dev/null
[ $? = 0 ] && echo "Started listener"
```

11. Shutdown and reboot both systems. Verify that the process called "listen" is running on *guardian*. You should now be able to issue **uucp** requests for *guardian* from *colossus*.

The nlsadmin(ADM) command

The **nlsadmin** command described in this section does the following:

- Creates directory */usr/net/nls/inet/tcp* (step 3A), and creates files called *dbf*, *addr*, and *lock* in this directory.
- Adds the following line to the *dbf* file in the directory */usr/net/nls/inet/tcp* (step 3B):

  ```
  101:n:uucp:reserved:NULL,:/usr/lib/uucp/uucico -r0 -iTLI -uuucp" #TLI UUCP
  ```

  See the comments in this file for an explanation of the fields in this entry.
- Adds the following line to the file *addr* in the directory */usr/net/nls/inet/tcp* (step 3C):

  ```
  \x020002ba8493904d0000000000000000
  ```

  Port numbers greater than 255 are split across two bytes. In our example, the port being used was 698. In the hexadecimal representation, this port was **02 | ba**, in two bytes. We do a byte by byte conversion to octal. **02** in octal is \002, and **ba** is \272, hence we have \002\272, not \001\272, which would be a direct conversion from decimal. Port numbers less than 256 should have the first byte in both hexadecimal and octal equal to 0. For example, port 253 is **00 | fd** in hexadecimal, and \000\375 in octal.

See **nlsadmin**(ADM) for more information.

Connecting two local systems using a direct wire

This section describes how to install a direct wire between two computers. 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 do the following:

- Choose a serial port on each machine.
- Connect a serial wire (RS-232) between the two machines, using the chosen serial ports.
- Edit the UUCP configuration files.

Choosing a serial port

On each machine, you must 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. If you remove a terminal, make sure no one is logged in.

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

You 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 vice versa. Because the connections can vary, check the hardware manuals for each computer to determine the proper pin connections.

Edit the UUCP configuration files

You should edit the UUCP files as instructed in "Editing the UUCP control files" (page 331), except you should use the keyword **Direct** instead of **ACU** in the *Systems* files.

Testing a connection

The following steps will enable you to use the **cu** command on the main 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 main machine:

   disable /dev/tty*nn*

   where ***nn*** is the serial port you are using.

To connect to the remote machine from the main 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. Consult "UUCP error messages" (page 369) for a list of messages.

Complete UUCP examples

This section includes two complete working examples of a UUCP system and the database files.

Example 1: system gomer

The following system (*gomer*) has:

- 1200 baud modem on *tty2B*
- direct connection to system (*poker* - configured for dial-in) on *tty2d* for call-out only
- three valid UUCP logins:

nuucp	The public login for email. No password required.
ubarn	The on-site login for system (*poker*).
*upay*4	The private login for email and file transfers.

All lines beginning with "#" are comments and are not required. Most examples are partial listings and may contain other entries. The modem answers at 1200 baud first and is set up for both call in and out.

NOTE The lines from */etc/passwd* are included here for information only. You must use the **sysadmsh**(ADM) Accounts ➪ User ➪ Create or Accounts ➪ User ➪ Modify selections to create or alter UUCP login accounts.

/etc/passwd

```
uucp:*:5:5:Uucp admin:/usr/lib/uucp:
nuucp::201:5:public:/usr/spool/uucplogins/nuucp:/usr/lib/uucp/uucico
upay4:*:202:5:private:/usr/spool/uucplogins/upay4:/usr/lib/uucp/uucico
ubarn:*:203:5:poker:/usr/spool/uucplogins/ubarn:/usr/lib/uucp/uucico
```

/etc/group

```
uucp:x:5:uucp,nuucp,ubarn,upay4
```

/etc/systemid

```
gomer
```

/etc/inittab

```
Se2B:23:respawn:/etc/getty -t60 tty2B 2
Se2b:23:off:/etc/getty tty2b m
Se2D:23:off:/etc/getty -t60 tty2D 2
Se2d:23:off:/etc/getty tty2d m
```

/usr/lib/uucp/Devices

```
# 300-1200 baud hayes 1200 baud modem.
# The Direct tty4b entry is for programming the modem.
ACU     tty2B  -  300-1200  dialHA12
Direct  tty2b  -  300-1200  direct
poker   tty2d  -  9600      direct
```

/usr/lib/uucp/Permissions

```
# Public uucp login for mail only.
# Can send mail, transfer files to/from uucppublic, and get
# a directory (ls) listing.
LOGNAME=nuucp     MACHINE=OTHER \
        COMMANDS=rmail:ls:uucp \
        READ=/usr/spool/uucppublic:/usr/tmp \
        WRITE=/usr/spool/uucppublic:/usr/tmp \
        SENDFILES=yes   REQUEST=yes
# Private uucp login for mail and file transfer.
# Only dingbat, ogre, grinch, ... can use this login.
LOGNAME=upay4     VALIDATE=dingbat:ogre:grinch:blitzen \
        COMMANDS=rmail:ls:uucp:who:uux \
        READ=/  WRITE=/ \
        NOREAD=/etc \
        SENDFILES=yes   REQUEST=yes
# Local trusted connection to gomer
# Only poker can use this login.
LOGNAME=ubarn     VALIDATE=poker \
        COMMANDS=ALL \
        READ=/  WRITE=/ \
        SENDFILES=yes   REQUEST=yes
```

/usr/lib/uucp/Systems

```
# local calls
dingbat Any ACU 1200 4444444 ogin:-BREAK-ogin:-BREAK-ogin:
  uubig word: wetrot
# long distance (evening calls only)
grinch Any1800-0700 ACU 1200 18888888 "" \r ogin:-BREAK-ogin:
  -BREAK-ogin:nuucp
uunet Any1800-0700 ACU 1200 17031111111 ogin:-BREAK-ogin:
  -BREAK-ogin:xytpq sword: grm5q
# systems that call in as nuucp (for mail) but NOT call out.
daboss Never
sales Never
guru2 Nevea
poker Never
ogre Never
blitzen Never
```

Example 2: system dingbat

The following system (*dingbat*) has:

- 2400 baud modem on *tty1A*
- two valid UUCP logins:

  nuucp The public login for email. No password required.

  uubig The private login for email and file transfers.

All lines beginning with "#" are comments and are not required. Most examples are partial listings and may contain other entries. The modem answers at 2400 baud first and is set up for both call in and out.

/etc/passwd

```
uucp:*:5:5:Uucp admin:/usr/lib/uucp:
nuucp:*:201:5:public:/usr/spool/uucplogins/nuucp:/usr/lib/uucp/uucico
uubig:*:202:5:private:/usr/spool/uucplogins/uubig:/usr/lib/uucp/uucico
```

/etc/group

```
uucp:x:5:uucp,nuucp,uubig
```

/etc/systemid

```
dingbat
```

/etc/inittab

```
Se1A:2:respawn:/etc/getty -t60 tty1A 3
Se1a:2:off:/etc/getty tty1a m
```

/usr/lib/uucp/Devices

```
# 300-2400 baud hayes 2400 baud modem.
# The Direct entry is for programming the modem.
ACU  tty1A  -  300-2400  dialHA24
Direct  tty1a  -  300-2400  direct
```

/usr/lib/uucp/Permissions

```
# Public uucp login for mail only.
# Can send mail, transfer files to/from uucppublic, and get
# a directory (ls) listing.
LOGNAME=nuucp     MACHINE=OTHER \
        COMMANDS=rmail:ls:uucp \
        READ=/usr/spool/uucppublic:/usr/tmp \
        WRITE=/usr/spool/uucppublic:/usr/tmp \
        SENDFILES=yes   REQUEST=yes
# Private uucp login for mail and file transfer.
# Only ogre, grinch, ... can use this login.
LOGNAME=uubig     VALIDATE=ogre:grinch:gomer:blitzen \
        COMMANDS=rmail:ls:uucp:who:uux \
        READ=/  WRITE=/ \
        NOREAD=/etc \
        SENDFILES=yes   REQUEST=yes
```

/usr/lib/uucp/Systems

```
# local calls
gomer Any ACU 2400 3333333 ogin:-BREAK-ogin:-BREAK-ogin:
  upay4 word: dryrot
# long distance (evening calls only)
grinch Any1800-0700 ACU 2400 18888888 "" \r ogin:
  -BREAK-ogin:-BREAK-ogin: nuucp
# systems that call in as nuucp (for mail) but NOT call out.
daboss Never
damgr Never
guru2 Never
ogre Never
blitzen Never
```

Sample commands

Send mail to another system and have it send the mail back.

mail ***othersystem*!*mysystem*!*mylogin*** (Bourne/Korn shell)

mail ***othersystem*\!*mysystem*\!*mylogin*** (C shell)

NOTE MMDF must be configured before you can send mail via UUCP.

Send the file */usr/spool/uucppublic/data* to another system.

uucp ~/data ***othersystem***!~/ (Bourne/Korn shell)

uucp \~/data ***othersystem***\!\~/ (C shell)

Display the systems you can call:

uuname

Force a call to another system and save the debug output in background:

/usr/lib/uucp/uutry -r -x9 ***system***

system is the name of the system you want to debug. The output is saved to a file with the same name as the system in *tmp* directory.

Administering your UUCP system

This section discusses the various shell scripts that supervise and maintain UUCP. Consult "Administrative programs" (page 326) for commands available to the system administrator. Included is an extended description of the */usr/spool/uucp* work directory and a special subsection on troubleshooting.

UUCP maintenance shell scripts

There are several aspects of system operation that are governed by shell scripts running as daemons:

- checking the UUCP directory for work (**uudemon.hour**)
- polling of sites that are passive or do not originate calls (**uudemon.poll2**)
- sending of status information to the UUCP administrator (**uudemon.admin**)
- cleaning of the UUCP spool directory (**uudemon.clean**)

These scripts must be set up by the system administrator. See **uudemon**(ADM) for complete instructions.

Generating log reports on usage: uulog

The **uulog** program displays log information on UUCP usage according to remote machine. All usage of the programs **uucp, uuto,** and **uux** are logged in special log files, one per machine.

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

The UUCP spool directory

The following is a comprehensive discussion of all files and subdirectories of the UUCP spool directory. These files are created in spool directories to lock devices, hold temporary data, or keep information about remote transfers or executions.

TM. (temporary data file)

These data files are created by UUCP processes under the spool directory (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***, that is used when there are several *D.* files created for a work (*C.*) file.

X. (execute file)

Execute files are created in the spool directory 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:

- requester's login and computer name
- name of file(s) required for execution
- input to be used as the standard input to the command string
- computer and file name to receive standard output from the command execution
- command string
- option lines for return status requests

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 the text listed in these messages. 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 13-4 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.

(Continued on next page)

Table 13-4 ASSERT error messages
(Continued)

Error message	Description or action
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.
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.

(Continued on next page)

Table 13-4 ASSERT error messages
(Continued)

Error message	Description or action
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.
CAN'T FORK	An attempt to fork and exec 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. What follows is a list of the most common error messages that can appear in these files.

Table 13-5 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.
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 "UUCP error messages" (page 369).
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.

(Continued on next page)

Table 13-5 STATUS error messages
(Continued)

Error message	Description or action
DEVICE FAILED	The open of the device failed.
WRONG MACHINE NAME	The called machine is reporting a different name than expected.
CALLBACK REQUIRED	The called machine requires that it calls your system.
REMOTE HAS A LCK FILE FOR ME	The remote site has a LCK file for your system. They could be trying to call your machine. If they have an older version of UUCP, the process that was talking to your machine may have failed leaving the LCK file. If they have the new version of UUCP and they are not communicating with your system, then the process that has a LCK file is hung. 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.
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.

Chapter 14

Setting up electronic mail

This chapter explains how to set up electronic mail on your system.

Electronic mail on the SCO UNIX system is handled by two utilities, the MAIL USER AGENT (MUA) and the MAIL TRANSPORT AGENT (MTA). The MUA is the program, such as **mail**(C), that allows users to send, read, and manage mail messages. The MUA transfers the message to the MTA, the group of programs that actually route and deliver messages to their destinations. The MTA on the SCO UNIX operating system is MMDF (*M*ultichannel *M*emorandum *D*istribution *F*acility).

MMDF provides users with transparent access to the different networks and related mail transport PROTOCOLS, through CHANNELS, regardless of the MUA. (A channel is the method, such as UUCP, used to deliver messages; a protocol is a set of rules for communicating over a network and includes standards for mail message formats.)

In addition, MMDF provides the system administrator with tools to monitor and customize MMDF. Using these tools, the system administrator can tune MMDF dynamically, modifying the behavior of these programs even while MMDF is running.

With MMDF, users can send mail on the local network or over larger area networks across the interconnected group of networks known as the "Internet". The Internet is otherwise known as ArpaNet or DARPA (*D*efense *A*dvanced *R*esearch *P*rojects *A*gency) Internet.

The version of SCO MMDF provided with your operating system was derived from MMDF-II Release 43 from the University of Delaware. This version provides additional features, such as a more robust locking mechanism that allows you to specify how to lock user mailboxes, and support for **sendmail** *.forward* files. This version also differs in the format of the hashed table database built by **dbmbuild**(ADM), which is nonstandard.

The system automatically configures MMDF for local (one system) mail delivery when you install your operating system. If you did not install the entire distribution, you should install the MAIL package now using the **custom**(ADM) utility. See your *Installation Guide* for information on installing packages with **custom**.

Chapter overview

The first part of this chapter explains how to configure MMDF for most sites using the simple configuration utility provided with your distribution. First, the chapter gives a basic overview of how MMDF works and covers some background information necessary for using the configuration utility. With this information, you can then fill out the MMDF configuration checklist provided. Then, this chapter shows you how to run the configuration utility to set up MMDF to exchange mail with other computers.

The configuration utility does not work for all site configurations, therefore the second part of this chapter describes the different configuration files, how MMDF uses them, and how to modify them to work with your configuration. This section also covers how to test and maintain the MMDF system.

If you do not plan to exchange mail with other machines via UUCP or TCP/IP, you do not have to run the configuration utility; MMDF is already configured to send mail on the local machine. However, if you want to reroute mail sent to special system accounts, such as *root*, the configuration utility provides an easy way for you to set this up. For more information, see "Redirect mail for root?" (page 389).

How MMDF works

This section gives you an overview of how MMDF processes and delivers mail on your UNIX system. Most of the concepts relate to mail traveling in both directions; you should read both the outgoing and incoming sections with this assumption.

Outgoing mail

Outgoing mail starts when the user invokes the MUA, such as **mail**(C), to compose a mail message. The MUA requires that the user specify a "To:" header when creating the message. Then, the MUA adds two other headers, "Date:" and "From:", when the user sends the message. These headers specify how MMDF sends the message through the system; the next section explains the format of these headers.

Mail headers

For a mail header to be correct, it must include these three lines in the following format:

```
From: sender
To: recipient
Date: Weekday Mon DD hh:mm:ss year
```

Here is an example:

```
From: fred@npr.COM (Fred Astaire)
To: ginger
Date: Wed Apr  3 12:21:23 PST 1991
```

In addition, most MUAs and mail submission programs, such as **mail**(C) and **execmail**, add extra header lines. For example, the MUA might add the "Message-Id" header. If the MUA does not add this header, MMDF adds it, as well as any "Received" headers. The user can also add other headers. For example, if the user specifies a carbon-copy recipient, the message header includes a "Cc:" line. MMDF allows these additional header lines, but does not use them; the recipient mail server and MUA handle these header lines.

In the example above, note the format of the address in the "From:" line. This is an example of a DOMAIN NAME. MMDF uses the domain name to determine how to route the message. Before you can configure MMDF with the configuration utility, you must understand how domain names work; "MMDF configuration checklist" (page 383) covers domains and the different types of domain names, including fully qualified domain names, in greater detail.

Figure 14-1 shows the path that the message takes through MMDF once you send your message with the MUA.

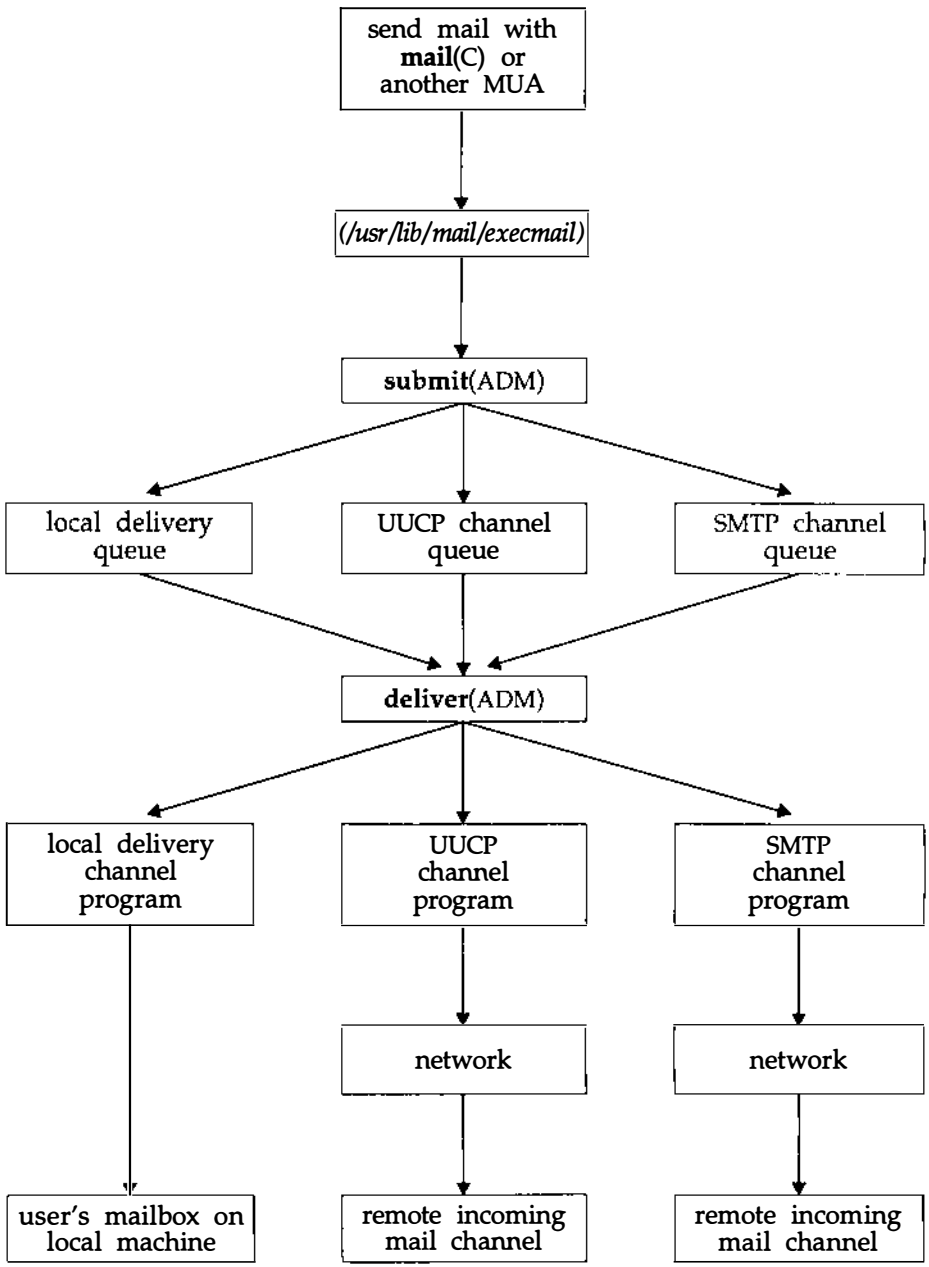


Figure 14-1 Outgoing mail route

The MUA allows the user to create, edit, and start a message on its journey. Then, the MUA transfers the message to **execmail**, which sends it to **submit**(ADM). The **submit** program uses the information in the "To:" line and the MMDF configuration files to build the fully qualified domain name. From this, **submit** determines the channel to use (for example, UUCP) when sending the message and then places the message in the appropriate channel queue for processing later by the **deliver**(ADM) program. The next time **deliver** runs (by default, **deliver** runs every 10 minutes), it transfers the messages from the channel queue to the appropriate channel program.

For example, you send a message with the following "To:" line:

```
To: andrei@npr.com
```

The **submit** program looks in the appropriate MMDF configuration table for *npr* and builds the fully qualified domain name; for example:

```
scribe.npr.com
```

Now, **submit** determines the appropriate channel to use to send the message. In this case, for example, all messages in the *npr.com* domain are sent to the outside world on the UUCP channel, so **submit** puts the message in the UUCP channel queue. Then **deliver** picks up the message and passes it to the channel program. The channel program transfers the message out of the MMDF mail system and into the UUCP subsystem where it is queued and sent via UUCP.

Incoming mail

MMDF processes incoming mail in much the same way it processes outgoing mail; Figure 14-2 illustrates how MMDF handles incoming mail.

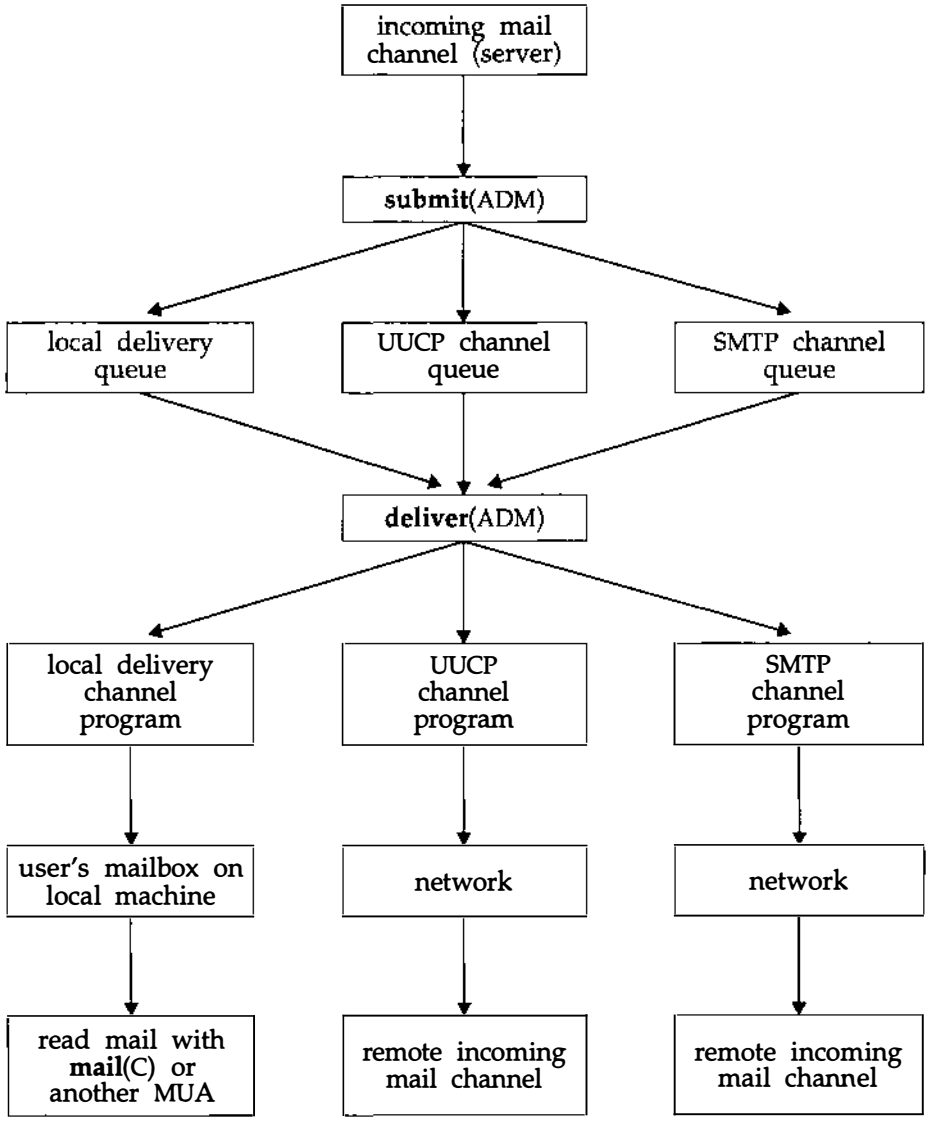


Figure 14-2 Incoming mail

MMDF processes incoming mail exactly as it processes outgoing mail (the same as when the MUA calls **submit**(ADM)) except that MMDF adds a "Received" line.

MMDF receives incoming mail from a channel's input program (called a "server"), such as **rmail**(C) for the UUCP program. The **submit** program determines the destination of the message and then designates the channel to use to deliver the mail. At this point, MMDF generally selects channels such as **local, badusers,** and **badhosts**. Mail destined for the local machine goes to the **local** channel; when the machine is local, but the user is not, mail goes to **badusers**; and if it requires further processing, the message goes to the **badhosts** channel. If the message is destined for another machine, such as when the local machine is acting as a gateway between networks, MMDF selects another network channel, such as UUCP.

The **submit** program puts the message in the appropriate queue and the **deliver** program transfers the mail from each queue to the appropriate channel; eventually the message is delivered to the user.

The configuration utility

The configuration utility facilitates MMDF configuration by building the appropriate configuration files at installation time. All sites should use the configuration utility to perform the initial MMDF configuration.

The configuration files that this utility builds are sufficient for most sites. If your site is one that the configuration utility does not handle, after building the initial configuration, you must edit the MMDF configuration files manually using the information in "MMDF configuration files" (page 404) and "Modifying MMDF configuration" (page 424). (See "Configuration utility limitations" (page 382) for a description of the limitations of this utility.)

In addition, if your configuration changes (for example, you add another machine to your network), you must rebuild the configuration files either by rerunning the configuration utility or by editing these files manually.

Configuring communications channels

Before you set up MMDF, you must configure all the communications channels that you plan to use to route mail. MMDF supports the following channels:

- UUCP (UNIX-to-UNIX Copy subsystem)
- SMTP (TCP/IP)

NOTE SMTP and TCP/IP channels form part of the TCP/IP product and are not supplied with the UNIX operating system.

The configuration utility configures for all the channels provided by SCO: UUCP, and SMTP. MMDF also supports other third-party mail products that provide channel programs; however, you cannot use the configuration utility to set up MMDF to use these products. See "Advanced MMDF configuration tasks" (page 436) for information on configuring MMDF for use with other channels. For example, if you plan to exchange mail with other hosts via UUCP, you must configure UUCP on the local host before running the configuration utility. For information on configuring UUCP, see the "Building a remote network with UUCP" chapter in this guide.

Configuration utility limitations

This section discusses the situations in which the configuration utility builds either an incorrect or incomplete MMDF configuration for your site. If the situation at your site matches one or more of the situations in this list, you must manually update the MMDF configuration files after building the initial configuration with the configuration utility.

- You use more than one communications channel to exchange mail with a particular host.

  If your host communicates with another host using more than one channel, the configuration utility sets up MMDF to communicate with that host using the first configured channel in the following order:

  1. TCP/IP
  2. UUCP

  If your host accesses another host using two or more channels, but you do not want to transfer mail using the first applicable channel, the configuration utility does not configure the MMDF correctly. See "Modifying MMDF configuration" (page 424) for more information.

- Your host does not exchange mail with every host on each configured channel.

  The configuration utility automatically sets up MMDF to exchange mail with every host on the configured channels. For information on removing hosts from a configured channel, see the sections on modifying the information for that channel in "Modifying MMDF configuration" (page 424).

- If you exchange mail with two hosts and you want to prevent one host from passing mail to the other host through the local host.

  The configuration script sets up MMDF to allow one host to send mail through the local host to the other host. If you want to prevent this, you must set up authorization to prevent the transfer. For information on how to do this, see "Configuring MMDF authorization" (page 437).

- You have more than one channel of the same type.

  The configuration utility lists all the hosts that your host accesses using a particular communications protocol in the same channel. To set up authorization to restrict a host's access on a channel, see "Configuring MMDF authorization" (page 437).

If your configuration changes for some reason after running the utility (for example, you might add a new UUCP host), you should reexamine this list. If the utility is still able to set up your site correctly, you can update your configuration by running the utility again; otherwise, you must update the configuration files manually.

MMDF configuration checklist

Once you configure the communications channels, use the checklist in this section to write down the information that you need to respond to prompts during the configuration procedure. Items in the checklist that are surrounded by boxes pertain to specific configurations (for example, if you are routing mail over UUCP); fill out these sections only if you have these configurations.

The sections following the checklist contain information on each of the items in the checklist.

Configuration Checklist

1. **Host name:** _____
2. **Domain name:** _____
3. **Fully qualified host name:** _____
4. **Hide your host name?** ☐ Yes ☐ No
5. **Redirect mail for *root*?:** _____
6. **Redirect mail for *mmdf*?:** _____
7. ***postmaster* address:** _____

If you have UUCP installed and configured and you plan to use MMDF to route mail over UUCP:

UUCP host name: _____

UUCP hosts with full domain names? ☐ No ☐ Yes; enter domain names:

_____

_____

_____

_____

If you have TCP/IP installed and configured and you plan to route mail over TCP/IP (SMTP):

Configure a name server? ☐ Yes ☐ No

8. **Smart host (badhosts):** _____
   (where to route mail to unrecognized machines)
9. **Smart host (badusers):** _____
   (where to route mail to unrecognized users)

Use the information in the following sections to complete the checklist.

Host name

The HOST NAME (or machine name) is the name of the machine on which you are configuring MMDF. To determine the host name, enter this command at your UNIX system prompt:

cat /etc/systemid

An example of a host name is *scribe.*

NOTE SCO MMDF, and MMDF in general, allow host names to contain underscore characters; this is an extension of internet standards. To ensure interoperability with hosts that strictly conform to internet standards, we recommend that you do not use the underscore character when internetworking is a primary goal.

Domain name

A DOMAIN NAME is the section of a mail address that appears to the right of the at (@) character, for example, *npr.com.* The domain describes the site where your machine is located and generally includes the machine (host) name, a department (optionally), and the site's organization or country. MMDF uses the domain to deliver the message to the appropriate location. Note that the domain name uniquely identifies a machine, but not the path by which messages reach that machine.

The following is the convention for specifying domains:

hostname.subdomain.top-level

If the domain includes a department, the convention is:

hostname.local.subdomain.top-level

Here is a description of each of the domain levels:

Top-Level Domain

A top-level domain is an officially registered name that describes the purpose of a group of institutions or a code that is associated with a country.

You can only use registered top-level and subdomain names if you have registered your organization with SRI International. For information on registering your domain, see "Registering domain names" (page 387). If you have not registered with SRI, use the UUCP top-level domain.

In the United States, the common top-level domains on the Internet are: *

COM	commercial institutions
code	country code †
EDU	educational and research institutions
GOV	government institutions
MIL	military institutions
NET	network
ORG	organization (generic)
UUCP	an unregistered domain name where users transmit information between cooperating neighbor machines via UUCP

Subdomain
An officially registered name that describes a company, department, or any subgroup under a top-level domain; *sco* is an example of a subdomain in the domain *COM*.

Local Domain
A name recognized only within an organization that has meaning only within that organization; a department name such as *engr* is an example of a local domain.

Fully qualified host name

The complete domain name (the machine name and all other domain names) is known as the FULLY QUALIFIED HOST NAME (or fully qualified domain name). For example, if you have this mail address:

```
andrei@scribe.npr.com
```

the fully qualified domain name is:

```
scribe.npr.com
```

In this case, *scribe* is the machine name, *npr* is the company, and *com* specifies that the machine belongs to a commercial organization.

* Domain names can be either upper or lowercase; MMDF is case-insensitive when evaluating domain names.

† The International Standards Organization (ISO) standard 3166 defines the country codes. For example, US is the country code for the United States, AU for Australia, DK for Denmark, and JP for Japan.

The following table shows examples of other fully qualified domain names:

slug.ucsc.EDU	*slug* is a machine at the University of California at Santa Cruz in the *EDU* domain
seismo.css.GOV	*seismo* is a machine at the Center for Seismographic Studies in the *GOV* domain
nessie.edinburgh.ac.UK	*nessie* is a machine at Edinburgh University in Scotland in the academic (*ac*) subdomain in the *UK* domain

To the configuration utility, the fully qualified host name is the host name followed by the local (if appropriate), subdomain, and top-level domain name as shown in the previous examples. The domain name refers to just the subdomain and top-level domain name (without the host name). For example, *npr.com.*

Registering domain names

If you or any user at your site plan to receive or send mail outside of your organization, you should register a top-level domain or subdomain with the NIC (Network Information Center). Even if you are only using UUCP, registration with the NIC is your only guarantee that the name of your site is unique.

To register a top-level domain and subdomain name call or write to the NIC standards organization at the following address:

Government Systems, Inc.
Attn: Network Information Center
14200 Park Meadow Drive
Suite 200
Chantilly, VA 22021
USA

Help desk telephone numbers are:

1-800-365-3642 (1-800-365-DNIC)
1-703-802-4535

Email address:

hostmaster@nic.ddn.mil
registrar@nic.ddn.nil, etc.

Help desk hours of operation:

7:00 am to 7:00 pm Eastern Time

Fax number:

1-703-802-8376

Network address:

192.112.36.5 (NIC.DDN.MIL)

Root domain server:

192.112.36.4 (NS.NIC.DDN.MIL)

The earlier your site enrolls a domain name and the NIC gives you an address, the less likely it is that you will have to alter a machine name or other site identifier later.

Hide your host name?

If you have several machines at your site (domain) and you register your domain name with the NIC, you can "hide" your host name behind your domain name. When you configure MMDF to hide your host name, people outside your organization can send mail to people who receive mail on different machines within your organization without having to know the name of those specific machines. In this case, MMDF identifies all outgoing mail as coming from a single source (your registered domain name). In addition, MMDF routes mail sent to this source to its correct destination within your organization.

For example, the fully qualified domain name for the host that you are configuring is:

```
scribe.npr.com
```

You can "hide" the host name *scribe* behind the domain name *npr.com*. In this case, someone outside the *npr* organization can send mail to *andrei* on *scribe* without specifying the fully qualified domain name. Instead, they can use *andrei@npr.com* as the address.

In addition, when *andrei* sends mail outside the organization, the message appears as if it came from *andrei@npr.com* instead of *andrei@scribe.npr.com*.

If you want to hide the current host behind the registered domain name, press ⟨Return⟩ when the configuration utility prompts you.

NOTE When you join local machines under a single domain name, you create an administrative domain. Within an administrative domain, all user names must be unique so that mail can go to any person anywhere within the domain without a local machine name in the mail address.

If you decide to hide the host behind a domain name, you should create ALIASES to map each user in the domain to the machine where they receive mail. For information on how to do this, see "Creating aliases for users" (page 402).

Redirect mail for root?

The system sends mail about any system problems to the *root* user. You can configure MMDF to redirect this mail to another person. For example, if you are the system administrator for a group of machines, you can redirect all mail sent to *root* on those machines to your own system mailbox. With this configuration, you do not need to log into each machine to read *root*'s mail.

If you want to redirect *root*'s mail, press ⟨Return⟩ when prompted by the configuration utility and then enter the address of the person to whom you want MMDF to deliver *root*'s mail. For example, if you are configuring MMDF on *scribe.npr.com* and you want the user *bob@talk.npr.com* to take care of mail to *root*, redirect *root*'s mail to *bob* by entering this at the prompt:

bob@talk.npr.com

(You can redirect *root*'s mail to a local user by entering just the user name at the prompt; you do not need to enter the fully qualified domain name for a local user.)

Redirect mail for mmdf?

When problems occur with the mail system, the system sends mail to the *mmdf* user. This account is reserved for administering your mail system; unless you log in as *mmdf* regularly, you might not find out about problems with the system. For this reason, it is a good idea to redirect *mmdf*'s mail to another person. To do this, press ⟨Return⟩ when the configuration utility prompts you and then enter the address of the person to whom you want to deliver *mmdf*'s mail.

Postmaster address

On the Internet, people generally send any inquiries about user and host names to the "postmaster" address in that domain. RFC (Request for Comments) 821/822 requires that every host provide this reserved postmaster mailbox. For these reasons, you should designate a user on the local system or within the domain as the postmaster and define a *postmaster* alias. To set up the alias while running the configuration utility, simply enter the name of the postmaster user.

Setting up MMDF for UUCP

If you are setting up the system to exchange mail with another system using UUCP, the configuration utility prompts you for some information about your UUCP hosts.

UUCP host name

The configuration utility prompts for the UUCP name for the host that you are currently configuring. The configuration utility gets this information from **/etc/systemid** (or **uname**); when you install the operating system, the installation prompts you for the system name. In cases where the UUCP name is different from the host name, you should enter the correct name when prompted.

UUCP hosts with full domain names

The configuration utility asks if any of the UUCP hosts have full domain names. This might be the case if your host is not on the Internet but your site has an agreement with another machine on the Internet to transfer your mail. When people on the Internet send messages to you, they use the ***user@machine.domain*** address format (instead of the ***machine!user*** UUCP format). You then use UUCP to connect to that machine and pick up your spooled mail.

For example, your machine *scribe.npr.com* connects to *slug.ucsc.edu* on the Internet. The machine *slug.ucsc.edu* is a UUCP host with a full domain name. In this case, enter "y" when the configuration utility prompts you for UUCP hosts and enter the host name and then the fully qualified domain name at the prompts.

Configuring MMDF to use a name server

Versions of MMDF included with SCO UNIX System V/386 Release 3.2 Operating System Version 4.0 and later include NAME SERVER support. A name server is a program running on the network that provides a central database of information, such as Internet addresses and the names of hosts on which people receive mail.

If you want the configuration utility to set up MMDF to use the name server, you must set up the name server before running the utility. See Chapter 4, "Configuring the BIND name server", in the *TCP/IP User's and Administrator's Guide* for details on setting up the name server.

NOTE The name server forms part of the TCP/IP product, not the UNIX operating system.

If you already have a name server running, the configuration utility sets up MMDF to use it automatically.

Configuring smart hosts

If the machine that you are configuring communicates directly with another machine that has more complete information about the entire mail network, you can set up MMDF to route any mail that it does not recognize to that machine. This machine is known as a "smart host". MMDF recognizes two kinds of smart host: hosts with information about machine names and hosts with information about the users on the network.

Badhosts channel

If you specify a smart host for the machine names, MMDF routes mail destined for machines that it does not recognize to the **badhosts** channel. All mail directed to this channel is sent to the smart host that you specify via the normal channel (such as UUCP) that you use to communicate with that host.

For example, someone sends mail to *boris@kgb.gov* but the local host does not recognize the machine *kgb.gov*. However, your host uses UUCP to communicate with another machine, *rocky.npr.com*, that has more complete information about the network. If you configure *rocky.npr.com* as your smart host, MMDF puts the message to *boris@kgb.gov* in the **badhosts** channel and then uses UUCP to deliver it to *rocky.npr.com*. The mail system on *rocky.npr.com* then determines the correct route to send the message to its destination.

Badusers channel

You can also specify a smart host that contains complete information about all the users on a network. This is useful if you have a large number of people at your site. Instead of maintaining information on all the people at the site on each host, you can maintain this information on one central host (the smart host); each individual host maintains information about the local users only. In this case, any mail addressed to users that the local machine does not recognize is directed to the **badusers** channel. The mail routed on this channel is sent to the smart host via the channel (such as UUCP) that you usually use to communicate with that host.

For example, someone on the local host sends mail to *natasha* but the host does not recognize her as a local user. If you communicate with *moose.npr.com*, a smart host that contains complete user information, MMDF routes the message to this host. This does not mean that *natasha* actually receives her mail on *moose*, just that *moose* has more information about where *natasha* is located.

Running the configuration utility

Once you configure the communications channels (and name server, if appropriate) and complete the configuration checklist, you are ready to run the MMDF configuration utility.*

NOTE The configuration utility provides common default values for many of the prompts; to accept these values, press ⟨Return⟩ at the prompt.

The configuration procedure varies slightly (from step 11 onwards) depending on whether TCP/IP is installed. The procedure detailed in this section assumes that TCP/IP is not installed. If TCP/IP is installed on your system, follow the instructions down to step 11 and then continue from the section headed "Running the configuration utility with TCP/IP installed" later in this chapter.

Use the following procedure to configure MMDF:

1. Change to the */usr/mmdf* directory and start the configuration utility by entering the following command as *root*:

   mkdev mmdf

   You see a screen with information about the limitations of the configuration utility.

2. At the bottom of the screen, you see this prompt:

```
Do you wish to continue the configuration process at this time? [y]
```

   If you enter "n" at this prompt, you exit the configuration utility and return to the UNIX system prompt. Press ⟨Return⟩ or enter "y" to continue with the configuration.

   The configuration utility displays a message about the current version of the software, for example:

```
This machine is running level 43NS MMDF.
```

* Remember, if you are only planning to route mail on the local machine, you do not have to run the configuration utility; MMDF is already set up for local mail delivery.

3. If you have not already set the host name on the machine that you are configuring, you see the following prompt (if you have already set the host name, skip this step):

```
The name of this host has not been configured yet.
What will you be calling this host?
```

   Enter the host name at this prompt. You must set the name of the host before configuring MMDF.

4. Now, you see the following prompt where ***host.domain*** is a fully qualified host name like *scribe.npr.com*:

```
Is your fully qualified host name 'host.domain'? [y]
```

   If the fully qualified host name is correct, press ⟨Return⟩ and go to the next step. If the host name is incorrect, enter "n". The utility displays this prompt:

```
What is the correct host name? [host]
```

   Enter the correct host name (for example, *pubsco*) or press ⟨Return⟩ to select the default host name. You see the prompt for the domain name:

```
What is the correct domain name? [domain]
```

   There are three possible responses to this prompt: you can press ⟨Return⟩ to accept the default domain name, you can enter a different domain name (for example, *eng*), or you can enter a domain and sub-domain name (for example, *techpubs.eng*).

   Now, the utility prompts you to confirm the fully qualified host name:

```
Is your fully qualified host name 'pubsco.techpubs.eng'? [y]
```

   If the host name is correct, press ⟨Return⟩ and go to the next step; otherwise enter **n** and repeat this step.

   If you have specified a fully qualified host name which includes a sub-domain, for example, *pubsco.techpubs.eng*, the prompt shown in step 5 is displayed; if you have specified a fully qualified host name which does not include a sub-domain, for example, *pubsco.eng*, step 5 is bypassed.

5. Now, you see some information about "hiding" the host behind the domain, followed by this prompt:

```
At many sites, it is common for mail to be addressed as being from
"person@site" instead of from "person@machine.site". This allows people
to be moved between machines internally without requiring them to notify
all their external correspondents about the address change. This
configuration does, however, require a complete user alias table (see
below) containing mappings from user names to the host that they actually
plan to read their mail on. If you are not sure, then you should
probably answer "yes" if there are two or more machines in the domain
"techpubs.eng", and "no" otherwise.

Do you wish to hide 'pubsco.techpubs.eng' behind 'techpubs.eng'? [y]
```

   To "hide" the host that you are configuring behind another host, press ⟨Return⟩ at this prompt. Otherwise, enter "n".

6. Now, the configuration utility displays some information about creating mail aliases for special accounts, *root*, *mmdf*, and *postmaster*, on your system.

   First, the utility prompts you to create an alias for *root*:

```
Do you wish to have mail for root redirected to a real user? [y]
```

   If you want mail addressed to *root* to go to *root*'s system mailbox, enter "n" and go to the next step.

   If you want *root*'s mail to go to a different person, press ⟨Return⟩ and you see this prompt:

```
To whom should root's mail be sent?
```

   Enter the login name of the person that you want to receive mail directed to *root*.

7. Now, you see this prompt:

```
Do you wish to redirect mail addressed to mmdf? [y]
```

   If you do not want to redirect mail to *mmdf*, enter "n" and go to the next step. If you want mail addressed to *mmdf* to go to another person (for example, the system administrator), press ⟨Return⟩.

```
To whom should mmdf's mail be sent? [root]
```

The default for this prompt is the same as the selection made for step 6. For example, if *root* was selected in response to step 6 then *root* is displayed here.

To have *mmdf*'s mail go to *root*'s mailbox, simply press ⟨Return⟩. If you want this mail directed to another person, enter the address of that person.

8. You see this prompt:

```
To whom should mail addressed to postmaster be sent? [mmdf]
```

Again, you can direct any mail sent to *postmaster* to another person, such as *mmdf*. Press ⟨Return⟩ to accept the default or enter a different address.

9. If UUCP is not installed on your system, you see a message like this:

```
UUCP not installed, skipping ...
```

10. Now, you see some information about associating login names for each user to the machines where the users actually read their mail by creating an alias file. See "Creating aliases for users" (page 402) for more information.

The configuration utility also displays some information about converting XENIX aliases to the correct MMDF format. For more information, see "Converting from XENIX" (page 428).

If you have UUCP installed on your system, you see this prompt:

```
Are you going to be using UUCP for mail? [y]
```

If you plan to use UUCP for transferring mail, press ⟨Return⟩.

NOTE If you are running the configuration utility with TCP/IP installed on your system, go to "Running the configuration utility with TCP/IP installed" (page 399).

11. Now, you see the prompt for the UUCP host name:

```
Is this host known as 'host' for UUCP? [y]
```

If the host name is correct, press ⟨Return⟩ and go to the next step. Otherwise, enter "n" to display the following prompt:

```
What is this host's UUCP name? [host]
```

Enter the new name and press ⟨Return⟩. You see this prompt again:

```
Is this host known as 'host' for UUCP [y]
```

Press ⟨Return⟩ if the host name is correct, or press "n" to change it.

12. At this point, if the configuration utility detects a UUCP connection to the *sosco* machine at SCO, you see the following prompt; if not, go to the next step.

```
uuname reports a connection to sosco.
Does this link really exist? [n]
```

If your system has a UUCP connection for exchanging mail with *sosco*, enter "y" and go to the next step. If not, enter "n" to display the following message:

```
Deleting sosco from list of mail sites
```

If *sosco* was the only configured UUCP connection, the utility exits and displays the following message:

```
UUCP must be configured before mail
```

In this case, configure your UUCP connections before running **mkdev mmdf** again.

13. Now, the configuration utility prompts you for information about the hosts you communicate with using UUCP:

```
Because UUCP does not maintain information about domain names, it will be
necessary for you to provide the domain names of any of the hosts with
which you communicate via UUCP.

Do any of your UUCP hosts have full domain names? [n]
```

In most cases, UUCP hosts do not have full domain names, so you can simply press ⟨Return⟩ at this prompt.

If any of your UUCP hosts have full domain names, enter "y". You see the prompts for the host names of your UUCP hosts:

```
Enter the UUCP site name (blank to terminate):
```

Enter the host name, for example, *pubsco* (not the fully qualified name) and press ⟨Return⟩.

14. The following screen is displayed:

```
What is pubsco's fully qualified name? [pubsco]
```

Enter the fully qualified name, for example, *pubsco.techpubs.eng* and press ⟨Return⟩.

15. You see the following messages:

```
Micnet not configured, skipping ...

TCP/IP not installed, skipping ...

Many sites do not have complete information about the entire mail network,
but rely on another "smarter" host to determine the correct route that
mail messages should follow to reach their destinations. Any mail that
the local machine is incapable of correctly handling is passed to the
smart host for further processing. In MMDF, this is called the "badhost"
channel.

Do you have such a "smart" host? [y]
```

If you enter "n", go to the next step; if you enter "y" the following screen is displayed:

```
What is its name? (q if you have changed your mind)
```

16. The following screen is displayed:

```
Another option, which is often used on large sites is to have a central
machine which contains complete knowledge about all the users on the site,
and only maintaining local lists on each machine. MMDF provides the
facility to forward mail containing unrecognised local addresses to a
smarter host which will have a complete user data base (via the "baduser"
channel).

Do you have such a "smart" host? [y]
```

If you enter "n", go to the next step; if you enter "y" the following screen is displayed:

```
What is its name? (q if you have changed your mind) [pubsco.techpubs.eng]
```

17. When you finish, the configuration utility displays information similar to the following as it creates and edits the MMDF configuration files (the messages might be different, depending on your configuration):

```
Creating the mmdftailor file: header, host name info, support
address, alias tables, local domain and channel, SMTP, root

Building the alias tables (mostly empty)
A Mailing list alias table (alias.list) already exists, skipping
A general user alias table (alias.user) already exists, skipping

Building channel files
local, list, SMTP, UUCP, badusers, badhosts

Building the domain tables
local, domain, root
done
building the database
```

The configuration is now complete.

Running the configuration utility with TCP/IP installed

This section assumes that TCP/IP is installed on your system. It continues on from step 10 in the previous section.

11. You see the following messages:

```
Micnet not configured, skipping ...

Are you going to be using SMTP for mail? [y]
```

If you enter "n" go to step 13; if you enter "y" go to step 12.

12. The configuration utility displays one of the following messages:

- If the name server is already configured on your system, you see the following screen:

```
A domain name server is running, mail will be configured to use it.
```

In this case, the configuration utility configures MMDF on the host that you are configuring to use the name server automatically.

- If the name server is not configured, you see:

```
The name server is not currently configured on this machine.
Do you plan to configure a name server on the local network? [n]
```

If you do not want to configure a name server, press ⟨Return⟩ to continue with the configuration utility.

If you plan to configure a name server, you must do so before configuring MMDF. In this case, enter "y". The configuration utility exits and displays this message:

```
Name server must be configured before mail.
```

Set up the name server and then run **mkdev mmdf** again. For information on configuring the name server tables, refer to Chapter 4, "Configuring the BIND name server," in the *TCP/IP User's and Administrator's Guide.*

13. The following screen is displayed:

```
Many sites do not have complete information about the entire mail network,
but rely on another "smarter" host to determine the correct route that
mail messages should follow to reach their destinations. Any mail that
the local machine is incapable of correctly handling is passed to the
smart host for further processing. In MMDF, this is called the "badhost"
channel.

Do you have such a "smart" host? [y]
```

If you enter " y " the following screen is displayed:

```
What is its name? (q if you have changed your mind) [ ]
```

14. The following screen is displayed:

```
Another option, which is often used on large sites is to have a central
machine which contains complete knowledge about all the users on the site,
and only maintaining local lists on each machine. MMDF provides the
facility to forward mail containing unrecognised local addresses to a
smarter host which will have a complete user data base (via the "baduser"
channel).

Do you have such a "smart" host? [y]
```

If you enter " y " the following screen is displayed:

```
What is its name? (q if you have changed your mind) [ ]
```

15. When you finish, the configuration utility displays information similar to the following as it creates and edits the MMDF configuration files:

```
Creating the mmdftailor file: header, host name info, support
address, alias tables, local domain and channel, SMTP, root

Building the alias tables (mostly empty)
A Mailing list alias table (alias.list) already exists, skipping
A general user alias table (alias.user) already exists, skipping

Building channel files
local, list, SMTP, UUCP, badusers, badhosts

Building the domain tables
local, domain, root
done
building the database
```

The configuration is now complete.

Preparing MMDF for use

Before you can use your mail system, you must perform steps to notify the UNIX system of the new configuration. The following sections contain information on restarting **deliver** and making this change permanent in the MMDF system startup file.

Restarting the deliver daemon

When you enter multiuser mode, the system automatically starts a **deliver** process for the local channel only. Each time someone on the system sends or receives a message, the **deliver** daemon runs, placing the message in the correct channel.

With the MMDF configuration utility, you configured MMDF to use additional channels. However, the configuration utility does not affect the **deliver** daemons that are currently running. To make the changes to your MMDF configuration take effect, stop and restart any **deliver** daemons using these commands:

1. As *root*, enter the following command:

   ps -ummdf | grep deliver

   This command displays any **deliver** processes running on your system. For example:

   ```
   285  ?         0:00  deliver
   ```

2. Stop the **deliver** process with **kill**(C). For example:

   kill 285

3. Now, restart **deliver** with this command:

   /usr/mmdf/bin/deliver -b

   This command restarts the **deliver** daemon so that it runs every 10 minutes. If you want **deliver** to run more often, use the **-T** option. We recommend that you set up **deliver** to run every 60 seconds using the following command:

   /usr/mmdf/bin/deliver -b -T60

NOTE When you restart the **deliver** daemon, the change is only temporary. The next time you reboot, the system reads the original **deliver** command in */etc/rc2.d/S86mmdf* and starts **deliver** for the local channel only. To make your changes permanent, modify the *S86mmdf* file using the information in the next section.

Modifying MMDF system startup

Unless you want to run multiple **deliver**(ADM) daemons, you do not need to modify MMDF system startup. The MMDF system startup file */etc/rc2.d/S86mmdf* already includes information for starting the **deliver**(ADM) daemon on all the configured channels. The system reads the *S86mmdf* file automatically when you enter **init** state 2 (multiuser mode).

By default, the line in this file looks like the following:

```
/bin/su mmdf -c "/usr/mmdf/bin/deliver -b"
```

To edit *S86mmdf*:

1. First, log in as *root*.
2. Edit */etc/rc2.d/S86mmdf*.
3. If you want **deliver** to run more often than the default of every 10 minutes, add the **-T** option on this line. For example, set up your system so that **deliver** runs every 60 seconds using the following **deliver** startup line:

   ```
   /bin/su mmdf -c "/usr/mmdf/bin/deliver -b -T60"
   ```

NOTE When you start the **deliver** program with the **-b** option only, *one* **deliver** process manages *all* the channels that you specified with the configuration utility. If you want different **deliver** processes to manage each of your channels or more than one **deliver** process for a single channel, refer to "Running multiple deliver daemons" (page 432).

Other common configuration tasks

Depending on your configuration, you might need to perform some additional MMDF configuration tasks after running the configuration utility and restarting **deliver**. This section covers some common configuration tasks; for additional configuration information, refer to "Modifying MMDF configuration" (page 424) and "Advanced MMDF configuration tasks" (page 436).

Creating aliases for users

If you chose to hide your local host name behind your domain name in step 6, you should add the login names of the users on the system and machine names where they receive mail to the */usr/mmdf/table/alias.user* file. The configuration utility creates *alias.user* for you; add information to this file using the following format:

user: ***user@machine.domain***

For example, to map *andrei* to the machine *scribe* in the *npr.com* domain, use this format:

andrei: andrei@scribe.npr.com

See "Alias files" (page 412) and "Maintaining user aliases" (page 425) for more information on setting up aliases on your system.

Converting XENIX alias files

If you are converting your mail system from XENIX to UNIX, you must use the **mmdfalias** utility to convert a XENIX-style */usr/lib/mail/aliases* file to the correct format for MMDF. For information on how to do this, see "Converting from XENIX" (page 428).

Changing logging levels

By default, the configuration utility sets the logging levels to log fatal errors only (FAT). If you are switching to MMDF from a **sendmail**-based system, you might want to change the **MMSGLOG** tunable parameter, in the */usr/mmdf/mmdftailor* file, from FAT to FST to retain a logging level equivalent to "mqueue/syslog". See "Logging levels" (page 411) and "Changing MMDF parameters" (page 449).

Rebuilding the MMDF hashed database

The configuration utility rebuilds the MMDF hashed database automatically after modifying the configuration files. However, if you modify the alias files or */usr/mmdf/mmdftailor* **after** running the MMDF configuration utility, you must rebuild this database manually.

To do this, enter the following commands as user *mmdf*:

```
cd /usr/mmdf/table
./dbmbuild
```

NOTE You must rebuild the MMDF database every time you make changes to the alias or routing files in the */usr/mmdf/table* directory.

MMDF configuration files

The configuration utility provided with your MMDF distribution builds configuration files sufficient for most sites, but not for every site; see "Configuration utility limitations" (page 382) for more information. In cases where the files that the configuration utility builds are insufficient, you must edit these files manually. This section describes the different configuration files, how MMDF uses them, and how to modify them to work with your configuration. This section also explains how to test and maintain the MMDF system.

The material presented in this section assumes that you have already set up the channels over which you plan to route mail and that you built the initial MMDF configuration using the configuration utility.

This section begins with an introduction to the different configuration files and formats that MMDF uses to route mail.

Table 14-1 briefly describes the MMDF configuration files.

Table 14-1 MMDF configuration files

File(s)	Description
mmdftailor	defines all the mail attributes for the local machine
alias files	defines aliases for mailing lists, programs, users and system accounts
domain files	matches host names to fully qualified host names
channel files	expands fully qualified host names to specify the addressing information necessary to reach the host or domain

The following sections describe the format of these files in detail. The **tables**(F) manual page also contains information about the file formats.

The mmdftailor file

The */usr/mmdf/mmdftailor* file defines all the mail attributes for the local machine, such as its name, the domain, channel, and alias files to use, how to set up each channel, and how to perform logging.

By default, MMDF is distributed with a simple *mmdftailor* file that is configured for local mail only. When you perform the initial MMDF configuration (as described in "Running the configuration utility" (page 392)) the configuration utility modifies the organization of the default *mmdftailor* file. This section describes the *mmdftailor* file generated by the configuration utility. Your *mmdftailor* file might be set up differently (for example, the names of your hosts are different and your *mmdftailor* file might not include all the channels in the example).

Table 14-2 briefly describes the keywords in the *mmdftailor* file; the sections that follow the table explain these keywords in more detail.

Table 14-2 mmdftailor file keywords

Keyword	Description
MLDOMAIN	defines the top-level domain name (for example, *COM* or *GOV*)
MLNAME	specifies the host name (*scribe*) or site name (*npr*) for hidden hosts
MLOCMACHINE	specifies the host name (for hidden hosts)
UUname	specifies the host name for UUCP
MSUPPORT	sets the address (*postmaster*) to send undeliverable mail and requests for information
MTBL	associates an abbreviated name with the more descriptive name for the alias, channel and domain files (also called "tables"). These abbreviated names are then used throughout the mmdftailor file as shorthand to refer to the table files.
ALIAS	defines the various sources for alias information, using the abbreviated names specified in the **MTBL** definitions
MCHN	defines the channels available to MMDF for mail transport*
MDMN	describes the domains known to MMDF
MMSGLOG	controls logging information from **deliver**(ADM) and **submit**(ADM)
MCHANLOG	controls logging that **MMSGLOG** does not control
MLCKTYPE	specifies the mailbox locking protocol

* MMDF requires at least one channel (**local**), for delivering local mail. See "Local mail configuration" (page 408) for more information.

The **MTBL**, **ALIAS**, **MCHN**, and **MDMN** keywords accept specific parameters; Table 14-3 briefly describes the parameters in the default *mmdftailor* file. See the **mmdftailor**(F) manual page for complete information about each of these keywords and their associated parameters.

Table 14-3 mmdftailor file parameters

Keyword	Parameter	Description
MTBL	name	specifies the shortname (table) for file
	file	describes the filename to associate with the shortname
	show	sets the description to display with programs, such as **checkup**(ADM)
ALIAS	table	specifies the name of the table to associate with the alias entry
	nobypass	prevents the *~address* alias bypass mechanism from working on the aliases in the specified table
	trusted	states that any alias in the described file is permitted to deliver mail to files or pipes using the permissions of any user on the system (including *root*)*
MCHN	show	sets the description to display with programs
	ap	specifies the type of address parsing used for the header of outgoing messages
	name	specifies the name of the channel
	mod	sets the delivery mode for the channel
	host	specifies the name of the host that is being contacted by this channel
	tbl	defines the file that lists the hosts that are accessible via the specific channel
	confstr	specifies channel-specific configuration parameters
	pgm	defines the channel program to invoke for this channel
MDMN	show	sets the description to display with programs
	table	specifies the name of the table containing information that describes the sites in this domain
	dmn	specifies the name of the domain

* Normally, because this permits such accessibility, you should only set the *alias* file (used for administrative aliases) to **trusted**.

MLDOMAIN

The **MLDOMAIN** keyword describes the top-level domain used by the local host. If the local machine has a registered top-level or subdomain name, the top-level domain name for your site, such as *COM* or *EDU*, appears in the **MLDOMAIN** line. If your host does not have a top-level domain, use UUCP. (UUCP is a generic name that MMDF uses for UUCP connections.) The **MLDOMAIN** line in the following example shows *COM* as the top-level domain:

```
MLDOMAIN com
```

MLNAME and MLOCMACHINE

Generally, **MLNAME** describes the name of the local host. In the following example, *scribe* is the host name:

```
MLNAME scribe
```

However, if you are "hiding" the local machine behind a domain name, **MLNAME** describes the subdomain name of your site (for example, *npr*) and **MLOCMACHINE** describes the name of the local host. (**MLOCMACHINE** is only used if you are hiding the local host.)

For example, if you are hiding the local host, *scribe*, behind the *npr* subdomain, these lines look like the following:

```
MLNAME npr
MLOCMACHINE scribe
```

For more information about hiding the host name, see "Hide your host name?" (page 388).

UUname

The **UUname** line describes the name used with UUCP; this definition must exist for UUCP to work properly. In most cases, this name is the same as the host name (in either **MLNAME** or **MLOCMACHINE**), but can be different if required. For example:

```
UUname scribe
```

MSUPPORT

The **MSUPPORT** line describes the address to send mail delivery problem notifications. If your site is connected to the Internet, you must define **MSUPPORT** as *postmaster*. On the Internet, people use this address to send any inquiries about user and host names in the domain. In addition, RFC821/822 (Request for Comments, an Internet standard) requires that every host on the Internet provide the reserved *postmaster* mailbox. Use the following line:

```
MSUPPORT postmaster
```

Note that the address you specify with **MSUPPORT** must be legal; if it is not and MMDF cannot deliver the original undeliverable mail to the support address, MMDF creates a new piece of mail that is undeliverable, and so on until the machine runs out of processes.

You can either create an alias in the *alias.ali* file to redirect *postmaster*'s mail to a user on the system, or create a *postmaster* account. For more information, see "Alias files" (page 412) and "Changing the postmaster alias" (page 425).

Alias configuration

The **MTBL** alias configuration keywords identify the filenames for the default alias files. By default, these **MTBL** lines look like the following:

```
MTBL name=alias,    file="alias.ali",    show="Administrative aliases"
MTBL name=lalias,   file="alias.list",   show="Mailing list aliases"
MTBL name=auser,    file="alias.user",   show="General user aliases"
```

Each **name** parameter defines the short name for the actual filename containing alias information (specified by **file**). Thus, the first **MTBL** entry identifies "alias" as the shortname for the *alias.ali* file. (MMDF uses this file to define the system administrative aliases for the local host.) See "Alias files" (page 412) for information about these files.

The **ALIAS** alias configuration keywords define additional information about the alias files in the **MTBL** lines. The following example shows the default **ALIAS** lines:

```
ALIAS   table=alias,    nobypass,   trusted
ALIAS   table=lalias,   nobypass
ALIAS   table=auser
```

NOTE The order of the alias files defined in this section determines the order that MMDF searches the files to find an alias.

For more information, see "Alias files" (page 412).

Local mail configuration

As with alias configuration, the **MTBL** keywords identify the filenames that contain local mail configuration information. The following example shows these table definition lines:

```
MTBL name=local,    file="local.chn",   show="Local Host Aliases"
MTBL name=locdom,   file="local.dom",   show="Local Domain"
```

In this example, the first **MTBL** entry identifies "local" as the shortname for the *local.chn* file. (The *local.chn* file contains information about the **local** channel on the local machine.) See "Channel files" (page 418) for more information about these files.

Below the **MTBL** definitions are the **MCHN** and **MDMN** definitions for local mail delivery. By default, the **MCHN** line for local delivery looks like the following:

```
MCHN   local,   show="Local Delivery",   ap=822,   mod=imm
```

The first parameter after **MCHN** is an arbitrary name that describes the channel (in this case, **local**). For a description of the **MCHN** parameters, see the **mmdftailor**(F) manual page. MMDF searches the channel tables in the order that the **MCHN** definitions appear in *mmdftailor*.

The local mail **MDMN** line looks like the following:

```
MDMN   "npr.COM",   show="Local domain",   table=locdom
```

The first parameter on the **MDMN** line specifies the name of the domain that the table lists (in this case, the local domain *npr.COM*).

List processing configuration

The list processing configuration section of *mmdftailor* defines the table and channel that MMDF uses to process mailing lists. By default, these lines look like the following example:

```
MTBL   list,   file="list.chn",          show="List Channel"
MCHN   list,   show="List Processing",   ap=same,   mod=imm,
       host="scribe.npr.COM"
```

The **MTBL** line defines the shortname for the file *list.chn*. This file contains information about passing mail addressed to mailing lists to the **list-processor** program (see "The alias.list file" (page 413) for more information).

SMTP configuration

If you configured your system to route mail over TCP/IP, your *mmdftailor* file contains **MTBL** and **MCHN** entries like the following:

```
MTBL   smtpchn, file=smtp.chn, show="SCO SMTP Channel"
MCHN   smtp, show="SCO SMTP Delivery", ap=822, tbl=smtpchn, mod=imm,
       confstr="scribe.npr.COM"
```

MTBL describes the SMTP channel file, *smtp.chn*. **MCHN** gives more information about the SMTP channel; the **mmdftailor**(F) manual page covers these parameters in more detail.

Local domain table configuration

The **MTBL** and **MDMN** entries in the local domain table configuration section describe the machines in the local domain. These lines look like the following:

```
MTBL domain,    file=domain.dom,       show="Local Ethernet"
MDMN "npr.COM", show="Local Ethernet", table=domain
```

The first parameter on the **MDMN** line specifies the name of the domain that the table lists (in this case, the *npr.COM* domain). The *domain.dom* file describes the machines located in the local domain.

UUCP configuration

If you configure MMDF to route mail over UUCP, the configuration utility adds parameters like the following that describe the UUCP configuration information:

```
MTBL uuchn,  file="uucp.chn",  show="NPR UUCP Channel"
MTBL uudom,  file="uucp.dom",  show="NPR UUCP Domain"

MCHN uucp,  show="NPR UUCP Delivery",  tbl=uuchn, ap=same
MDMN "UUCP",  show="UUCP Domain",  table=uudom
```

The two **MTBL** entries define the UUCP channel (*uucp.chn*) and UUCP domain (*uucp.dom*) files. The **MCHN** line describes more information about the UUCP channel and **MDMN** defines the file that describes the machine accesses via UUCP.

The badhosts channel

If you specified a "smart host" to redirect mail destined for machines that the local host does not recognize, the **MCHN badhosts** line looks like this:

```
MCHN badhosts, show="Last-Chance Routing", pgm=smtp, tbl=smtpchn,
     ap=822, host="rocky.npr.COM", confstr=scribe.npr.COM
```

The **MCHN host** parameter describes the smart host (in this case, *rocky.npr.COM*). In this case, the local host accesses the smart host via SMTP. The **pgm** parameter specifies the filename of the channel program (*smtp*) in the */usr/mmdf/chans*) directory and **tbl** specifies the file (*smtp.chn*) that describes the machines that the host accesses with SMTP.

The **badhosts** is not really a channel because it is not associated with its own transport program. In this example, the pseudo-channel uses the SMTP channel to relay mail to a more intelligent host. If the **badhosts** channel does not exist, MMDF returns mail to an unknown host to the sender.

The badusers channel

If you specified a "smart host" to redirect mail to users that the local host does not recognize, the **MCHN badusers** line looks like this:

```
MCHN badusers, show="Last-Chance Routing", pgm=smtp, tbl=smtpchn,
     ap=822, host="moose.npr.COM", confstr=scribe.npr.COM
```

The **MCHN host** parameter describes the smart host (in this case, *moose.npr.COM*). See the previous section for information about the **pgm** and **tbl** parameters.

The root domain table

The **MTBL** line defines the *root.dom* file which contains any domain information not named in other domain files.

```
MTBL rootdom, file="root.dom", show="Root Domain"
MDMN "", show="Root Domain", table=rootdom
```

The **MDMN** entry defines the name of root domain. (The root domain definition has no name ("") because the root domain file (*root.dom*) can contain entries for many different domains.)

Logging levels

By default, the configuration script sets **MMSGLOG** and **MCHANLOG** to the FAT logging level; this level logs fatal errors only. These entries look like the following:

```
MMSGLOG  level=FAT
MCHANLOG level=FAT
```

You can change these logging parameters to a higher level, such as FST to log full statistics, or add different logging controls, such as **AUTHLOG** to control authorization information. See the section on **MCHANLOG** in the **mmdftailor**(F) manual page for a list of levels. "Changing MMDF parameters" (page 449) also contains information about these logging levels. For more information about the MMDF logging files, see the **logs**(F) manual page.

Mailbox locking style

By default, MMDF uses the standard System V **fcntl**( ) kernel file locking protocol to lock users' mailboxes. However, if users on the system use MUAs that do not use the default locking protocol, you can configure the locking type with the **MLCKTYPE** keyword. The default **MLCKTYPE** line looks like the following:

```
MLCKTYPE advisory
```

See "Changing MMDF parameters" (page 449) for more information on setting locking protocols.

Alias files

An *alias* is an abbreviated name that MMDF translates into a larger string (a mail address or list of addresses). Aliases are useful for specifying a single name to represent a group of users. You might want to create an alias called *sales* to represent all the members of the Sales department in the company. To do this, you define an alias file in the */usr/mmdf/mmdftailor* file and then create the alias file in */usr/mmdf/table*. In the alias file, the *sales* alias entry might look like the following:[1]

```
sales:  joe, jane, bob, mike, karen, ann
```

When users want to send mail to everybody in the Sales department, they can use the *sales* alias like this:

mail sales

This sends a copy of the message to everyone in the Sales department.

You can name your alias files anything you like; however, you must define the file names in the */usr/mmdf/mmdftailor* file (see "The mmdftailor file" (page 404) for details). By default, MMDF provides the files listed in Table 14-4 that you can use to specify aliases for user names:

Table 14-4 MMDF alias files

File	Description
alias.list	aliases for lists of users
alias.user	aliases mapping users to their "home" host machines (the machines on which they receive their mail) and nicknames for local users
alias.ali	local machine system administrative aliases, programs, or files. At the minimum, this file should identify the aliases for *mmdf* and *postmaster*.

These files are located in the */usr/mmdf/table* directory.

1. The colon (:) following the alias name is optional.

NOTE You should *not* create aliases for remote users (users not located at the local site) in the alias files. In other words, do not create an alias like the following, where *moocow.uucp* is not a local machine at your site:

```
david: david@moocow.uucp
```

In this case, if a remote user sends mail to *david@npr.com*, the message is delivered to *david@moocow.uucp*.

In general, users should set private remote-user aliases using their MUA (for example, use the **alias** command in **$HOME**/*.mailrc*). For more information, about creating private aliases, see the chapter on electronic mail in the *User's Guide*.

The alias.list file

Use the *alias.list* file to create multiple-user aliases. With list aliases, MMDF processes the mail using the **list** channel and it appears as being from the sender instead of from the first person in the "To:" line.

To use the **list** channel, you must specify the following three lines for each alias:

- the name of the list
- the "***name***-outbound" line that contains the logins that comprise the list
- the "***name***-request" line that provides a login of the list maintainer who makes additions and deletions to and from the list

The following example shows how to use these three lines:

```
writers:              writers-outbound@list-processor
writers-outbound:     hanna,dianna,george,laurie,meg,naomi,steve
writers-request:      hanna
sales:                sales-outbound@list-processor
sales-outbound:       joe,jane,mike,karen,ann,uksales
sales-request:        joe
```

The **list-processor** keyword is a reserved word in MMDF and indicates access to the **list** channel for processing mailing lists. The *sales-request* alias provides a way to request additions or deletions to the *sales* alias.

When defining an alias that contains many user names, you can use a backslash character (\) as a line-continuation character. Use quotation marks ("") to delimit a string containing spaces or punctuation. When using an alias to define another alias, be careful not to create an alias loop.

You can also use the output redirection symbol (>) with the pipe character (|) to do more complex processing such as redirecting messages to files. For example:

```
Loguucp:        "network//usr/spool/log/uucp"
Logmlog:        "network|cat -v >>/usr/spool/log/mlog"
printer2:       "network|/usr/bin/lp -dprinter2"
```

In this example, MMDF pipes mail addressed to *Logmlog* to the **cat**(C) command to log the mail in the *mlog* file. MMDF pipes mail addressed to *printer2* to the **lp**(C) command for printing. These redirection alias examples use the user and group IDs of the user *network*. Although *network* is appropriate in most cases, you can specify any user named in the */etc/passwd* file on your system.

If you have a long list of names for an alias, you might want to include them in another file (instead of listing them directly in the *alias.list* file). To do this, use the **:include** keyword. For example:

```
staff:          staff-outbound@list-processor
staff-outbound: ":include:/etc/alias/staff"
staff-request:  ross
```

In this example, the **:include:** line specifies that you want to use the names listed in the */etc/alias/staff* file to define the alias. Note that you cannot use the backslash character (\) as a line-continuation character for lists of names in an **:include** alias list.

For details on setting up mailing lists, see the **list**(ADM) manual page.

The alias.user file

To map users to specific machines, specify aliases in the *alias.user* file. Use this file when you want each person at your site to receive mail on a particular machine. The following example shows how to set up *alias.user*:

```
andrei: andrei@scribe
george: george@dera
hanna:  hanna@scribe
karen:  karen@guardian
```

The alias.ali file

Use the *alias.ali* file for aliases that are not specific to *alias.list* or *alias.user*. This file usually contains aliases related to system administration, such as:

```
root:           david
mmdf:           david
postmaster:     david
uucp:           david
```

In this example, all mail addressed to *root*, *mmdf*, *postmaster*, and *uucp* is redirected to *david* on the local machine.

In general, do **not** create aliases for remote users (users not located at the local site) in the *alias.ali* file.

How MMDF uses alias files

When mail is addressed to *postmaster*, MMDF routes the mail by first searching the hashed alias table from *alias.ali* to expand the *postmaster* alias to the associated user name. For example, the *postmaster* entry in the *alias.ali* file:

```
postmaster:     david
```

Then, MMDF searches the *alias.user* file to find the local machine name associated with the user name. For example, the *alias.user* file contains an entry like this:

```
david:  david@golem
```

MMDF then uses this information when searching the various *.dom* files, which map the local machine name to a fully qualified host name and the *.chn* files, which map the fully qualified host name to information on how to route the message. "How MMDF routes mail" (page 420) explains the process for searching configuration files further.

Domain files

The domain files are used to match a host name to its fully qualified host name. Domain files serve two purposes: to convey information to MMDF about how machines are connected, and to specify special routing considerations for subdomains or top-level domains.

Domain files are named for the domain that they describe (except for *root.dom*, which contains domain information not named in other domain files); the filenames end in *.dom*. The general practice is to have a separate domain file for the domain in which the host machine resides.

The operating system distribution includes four domain files in the */usr/mmdf/table* directory. Table 14-5 lists these files.

Table 14-5 MMDF domain files

Domain	Domain file	Describes
local	local.dom	local machine
domain	domain.dom	machines in the local domain
uucp	uucp.dom	machines in UUCP domain
root	root.dom*	domains not listed in other domain files

* The *root.dom* file also contains information about how to access top-level domains, such as *MIL* and *GOV*. (The name *root* implies "top-level", as in a hierarchy.)

You can create new domain files for each domain. For example, create a domain file for the *npr.COM* domain and name it *npr.dom*. You do not need to use the **.dom* naming scheme; however, this filename extension makes it easier to determine the purpose of the file.

Domain file format

Each domain file consists of two columns of information: the left column lists the host name and the right column lists the fully qualified name for that host. The domain names can be either upper- or lowercase. Use tabs, spaces, a colon, or a combination of these characters to separate the first column from the second.

The name of a domain file determines the domain names for which MMDF searches. For instance, the domain file for UUCP generally contains entries for names in the following form:

```
machine:        machine.UUCP
```

However, you can create an entry to map a specific UUCP address to another address. For example, to map *research.UUCP* to *research.jcn.com*, the entry looks like the following:

```
research:       research.jcn.com
```

The following four sections give examples of different domain files.

The local.dom file

The *local.dom* file describes the local host. For example, the contents of the *local.dom* file on the host *scribe* in the domain *npr.COM* look like the following:

```
scribe: scribe.npr.COM
```

The domain.dom file

The *domain.dom* file describes the machines in the local domain (independent of the channel that MMDF uses to reach each machine). For example, if there are four machines in the domain *npr.COM*, the *domain.dom* file looks like the following:

```
scribe: scribe.npr.COM
huey:   huey.npr.COM
dewey:  dewey.npr.COM
louie:  louie.npr.COM
```

Note that the *domain.dom* file maps each machine to the fully qualified host name in the *npr.COM* domain.

The uucp.dom file

The *uucp.dom* file specifies the hosts in the UUCP domain. For example, if your host connects to the remote machine *cactus* via UUCP, the *uucp.dom* file looks like the following:

```
cactus: cactus.UUCP
palm:   palm.UUCP
```

In this case, MMDF directs any mail sent to the *cactus* or *palm* machines to the UUCP network.

The root.dom file

The *root.dom* file defines the hosts and domains not defined in the other domain files. For example, if your host connects to the UUnet network system, you can set up MMDF to send all mail to specific domains to *uunet.UU.NET*. To do this, set up your *root.dom* file like the following:

```
COM:    uunet.UU.NET
EDU:    ucscc.EDU
MIL:    uunet.UU.NET
GOV:    star.GOV
NET:    uunet.UU.NET
```

In this example, all mail directed to the *COM, MIL* is sent out on UUnet; mail to *GOV* goes to *star.GOV*; and mail sent to the *EDU* domain goes to *ucscc.EDU*.

LAN considerations

If you are configuring MMDF for use on a local area network (LAN), you can use the domain files to distribute the processing load on the machines, or you can designate a special mail server machine to route all the messages.

You have the following choices:

- MMDF Server — Designate one machine as the network server. (This machine may also have outer world access.)

  Each machine's domain file only needs to include **badhosts** and **badusers** channels that contain the fully qualified host name of the server. The domain file can be the same for each machine in the network. The network can grow, and machines can be added and removed with no effect on the domain files of other computers. The disadvantage is that the server receives a great deal of traffic and should be dedicated to its task. In addition, if the server is down, so is all electronic mail between machines. The best policy is a system of machines grouped around a server that has knowledge of another server; each server has knowledge of yet another and so on.

- Distributed Processing — Give each machine's domain file knowledge of each other machine in the network.

  The advantage is that networked machines can operate independently of each other. One machine's crash has no effect on the mail capability of the others (unless that machine is connected to another network via UUCP). The disadvantages are that system administration gets geometrically more difficult as you add or remove machines to or from the network. When a machine or user is added to or removed from the network, you must update all domain or alias files to recognize the change. Because a domain file can contain redundant information about the local machine, you can use one domain file on every machine. You should only use distributed processing for small networks.

  You can greatly simplify the distributed processing configuration by running a name server program on your network. For details on setting up the name server, see Chapter 4, "Name Server Operations Guide for BIND," in your *TCP/IP Administrator's Guide.*

- Gateways — Use a gateway to connect a network to another Local Area or Wide Area network. (This setup is derived from the server setup.) In this case, the server machine is typically connected to more than one network.

  In addition to containing the names of the local machines on the LAN, a gateway machine also contains the names of the other machines to reach over the other networks. This information is kept in the respective domain and channel files for the other networks and also in the *root.dom* file on the gateway machine. Other machines on the LAN use the **badhosts** channel to route non-local mail to the gateway machine, or a *root.dom* file that lists all the top-level domains as routing through the gateway. To avoid overloading the gateway machine, the other machines on the LAN use local domain files as described in the earlier "Distributed Processing" bullet.

Channel files

MMDF uses the channel files to determine the channel to use for outgoing mail and the address of the host on that channel. Channel files map the fully qualified host name (as determined from the domain file entries) to channel-specific addressing information. For example, the UUCP channel file maps host names to UUCP paths (using exclamation points) specifying how to get to each host.

The operating system distribution includes four channel files in the */usr/mmdf/table* directory. Table 14-6 lists these files.

Table 14-6 MMDF channel files

Channel	Channel file	Describes
local	local.chn	local machine
list	list.chn	list-processor
smtp	smtp.chn	machines accessed via SMTP
uucp	uucp.chn	machines accessed via UUCP

The **MCHN** definitions in the */usr/mmdf/mmdftailor* file direct MMDF to search the specified *.chn* files in the */usr/mmdf/table* directory for channel definitions.

Separate the left and right columns in the channel files by a space or tab, a colon character (:), or both.

The local.chn file

The *local.chn* file contains entries describing all the names local host is called, mapping them to the local host name. For example, if the local host is *scribe.npr.COM*, then the local host is known as *scribe* on the local machine. The *local.chn* file maps the local host name (on the right) to the different ways people might refer to *scribe*. This file looks like the following:

```
scribe:          scribe
npr.COM:         scribe
scribe.npr.COM: scribe
```

The list.chn file

The *list.chn* file contains information about the **list-processor** program:

```
list-processor: list-processor
list-proc:      list-processor
```

The left column is a pseudo-host defined in a mailing list alias (see "The alias.list file" (page 413) for details). These entries tell MMDF to pass mail addressed to a mailing list to the **list-processor** program.

The uucp.chn file

The *uucp.chn* file contains entries describing the hosts that your host connects to using the UUCP channel and how to route mail to those hosts. For example, the format of this file looks like the following:

```
mcvax.UUCP:      uunet!mcvax!%s
sri-nic.ARPA:    uunet!sri-nic.arpa!%s
uunet.uu.NET:    uunet!%s
```

The left column contains the UUCP host name from the domain tables; the right column describes the UUCP address that MMDF uses to direct mail to that host. The "%s" at the end of the UUCP address means to use the rest of the address from this point on. In other words, when mail is addressed to the user *hillis* at *mcvax.UUCP*, the UUCP channel passes the mail to UUnet along

with the rest of the UUCP address (*mcvax!hillis*). The second entry in this example shows how a domain name (*sri-nic.ARPA*) can be used within a UUCP path.

Channel file entries for the UUCP channel (in the *uucp.chn* file) when the destination machine is multiple hops away, appear as follows:

```
stooges.UUCP:   moe!curly!larry!stooges!%s
```

Specify the ***address*** of the host on the right hand side, where the ***address*** is a UUCP path.

The smtp.chn file

The *smtp.chn* file describes the hosts that you connect to using TCP/IP and the IP addresses of those hosts. The format of this file looks like the following:

```
cocoa.npr.COM:          123.456.789.1
caramel.npr.COM:        123.456.789.2
taffy.npr.COM:          123.456.789.3
```

The left column contains the fully qualified host names for the hosts that you connect to with TCP/IP and the right column contains the IP addresses. For more information about configuring MMDF to route mail over TCP/IP, see the *TCP/IP User's and Administrator's Guide.*

Following the pattern of mapping the host name in the left column to the addressing information for delivering to that host in the right column, you can create channel files for each **MCHN** definition in *mmdftailor*. The exception to this is the **badhosts** pseudo-channel definition; the **badhosts** channel program is determined at configuration time automatically. However, you can set up your channel files to indicate the channel that you use to reach the "smart host".

How MMDF routes mail

This section describes how MMDF uses the information in the configuration files to route mail on your system. Note that MMDF never searches the alias, channel, and domain files directly. When you build the hashed database with **dbmbuild**(ADM), the contents of these files are stored in tables in the *dbm* database. MMDF then uses the information in this database to route mail.

Mail arrives at and leaves computers using one of several different methods (such as UUCP, or TCP/IP called "channels". The MMDF **submit**(ADM) command accepts the incoming mail from a channel and determines the correct outgoing channel to use based on the destination host. The **submit** program uses the information in the domain tables to map the way the incoming mail describes the destination host to the way the host recognizes that destination

host. Based on the host description, **submit** uses the channel tables to determine the outgoing channel to use to route the message and places the message in the appropriate queue.

Then, using the information from */usr/mmdf/mmdftailor*, **deliver**(ADM) moves the mail from the queue to the appropriate channel. For example, if **submit** places a message in the UUCP channel queue, **deliver** moves that message to the UUCP channel. The **deliver** command can also place mail in a channel and let another program (such as **uux**(C) for the UUCP channel) carry out additional steps to resolve the circumstances dictated by the type of channel. The channel program sends the mail across the network to the proper destination.

Searching MMDF domain tables

The **submit**(ADM) command uses the domain tables for two purposes: to specify the fully qualified host name and (optionally) to specify the route to a host by listing the fully qualified host names of one or more intermediate hosts through which mail is to be routed.

First, **submit** separates the fully qualified host name into two parts: the name of the domain table and the host name to match on the left-hand side (LHS) of the entries in the domain table. For example, in the address *david@engr.canada.COM*, the name of the domain table to search is *canada.COM* and the host name to search for is *engr*.

MMDF tests an address for matches against the domain names in the **MDMN** entries in *mmdftailor*. For example, the address *david@engr.canada.COM* matches the following **MDMN** entry:

```
MDMN   "canada.com", show="Canada Delivery", tbl=canadadom
```

Then, MMDF searches *mmdftailor* for the **MTBL** entry for *canadadom*:

```
MTBL  canadadom,  file="canada.dom",   show "Canada Delivery"
```

MMDF uses the file associated with *canadadom* (*canada.dom*).

Thus, to route our message to *david@engr.canada.COM*, **submit** uses the following algorithm to search the domain tables for a match in the address:

1. Searches for the host name (*engr*) in the LHS of the domain table.
2. Searches in the LHS of the domain table (in this case, *COM*), if it exists, for the domain name (*engr.canada*).
3. Searches the LHS of the *root* domain table for the fully qualified host name (*eng.canada.COM*).

4. Searches the relevant tables that include ***flags=route***[2] in the **MTBL** line in *mmdftailor* for substrings of the address. For example, search the *COM* table for *canada*; search the *root* table for *canada.COM* or *COM*.
5. Searches the tables that include ***flags=partial***[3] in the **MTBL** line in *mmdftailor* for the input name, in this case, the fully qualified host name (*eng.canada.COM*), regardless of the domain name.

If **submit** finds no match at all, as a last resort, it uses the **badhosts** channel, if it exists. Because MMDF uses the first domain that has an exact match without looking for other matches in later tables, the order in which you list **MDMN** definitions is significant. Make sure the local domain table appears first and the *root* domain table is last in *mmdftailor*.

If the address provided is ***username@host*** (for example, *david@engr*), MMDF performs the same search as described above except that there is no *canada.COM* to match in the **MDMN** entries in *mmdftailor*. In this case, MMDF searches the *root* domain table and then the tables that include ***flags=partial*** in the **MDMN** line.

Delivery channel programs

A channel is a compiled program that permits a machine to talk to a single type of network communications protocol. Some simple channels only store mail for further processing. Channel programs reside in the */usr/mmdf/chans* directory.

Channel programs handle the communications protocol so that neither the operating system nor the rest of the MMDF system has to know about the intricacies of a particular communication protocol. This handling of the protocol is one of the advantages of MMDF; having specific protocol program modules permits a site to upgrade to other network types without having to rewrite the mail delivery system.

2. The ***flags=route*** parameter in the **MTBL** entry enables that table entry to match addresses for an entire subdomain, acting as a gateway. Do not use ***flags=route*** on tables other than the *root* domain table unless you have internal subdomains.
3. Use the ***flags=partial*** parameter on the *local* domain table so that users do not have to specify the full domain to send mail on the local machine.

The different channels are:

badhosts Called when a specified machine is unknown to the local machine. The use of "bad" is really a misnomer; any mail to an unknown machine is sent on this channel. You can assign **badhosts** in the *mmdftailor* file to a channel type such as **uucp** or **smtp**.

badusers Called when mail arrives at the local machine, but the user does not have a login on this machine, nor is there an alias for the user on the local machine. Generally, MMDF queues up this mail for submission to another machine with a larger list of users (a "smart host").

list Called to remail messages. This channel simply invokes **submit** and feeds the addresses and text back into the MMDF mail system. This is often used to avoid long address validation or to force the validation to occur in the background for very large mailing lists. This also ensures that MMDF sends any problem reports to the list maintainer.

local Called to deliver mail to mailboxes and processes on the local machine.

smtp Called to deliver or accept mail from a TCP/IP network connection. The **smtp** channel transfers messages by establishing a TCP/IP connection to a remote machine, and using the Simple Mail Transfer Protocol (SMTP) to send one or more mail messages. The Internet Protocol (IP) allows many local- or wide-area networks to be interconnected transparently. This permits the MMDF SMTP channel to exchange messages with any machine on any network to which it is connected. For example, if your machine connects to the Internet, you can exchange messages directly with any machine in the world that is also connected to the Internet.

uucp Called to direct mail to UUCP delivery to another machine, or to accept mail from a UUCP connection from another machine. Incoming mail is converted into the format specified by the Internet technical bulletin, RFC822, available from the DDN Network Information Center, or NIC.[4] Outgoing mail includes a "From<space>" line and the mail path arguments are separated by UUCP exclamation point characters (!).

For more information about UUCP, see the "Building a remote network with UUCP" chapter of this guide.

Channels act not only as protocol handlers, but in some cases actually initiate the communications to the network or to another machine as needed.

4. See "Registering domain names" (page 387) for more information.

For each channel, MMDF provides two programs: an input program and an output program. A channel program is associated with outgoing mail; a server program is associated with incoming mail. For example, the UUCP channel has */usr/bin/rmail* as its input program and */usr/mmdf/chans/uucp* as its output program.

On the MMDF system, the server is a channel program that monitors the network for incoming mail. On systems other than MMDF, the function of a mail server is built into the mail delivery system. Figure 14-3 describes the relationship between a channel and server.

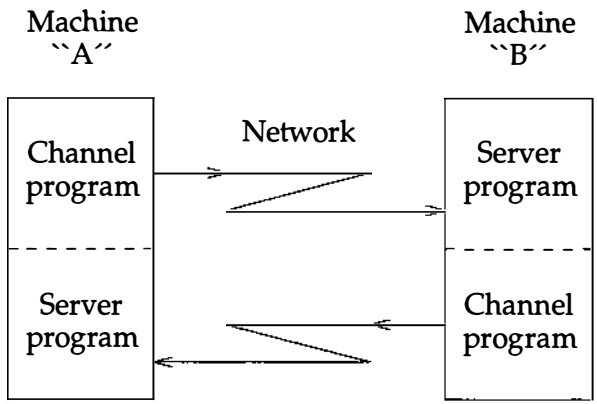


Figure 14-3 Channel and server relationship

Even if MMDF is only being run on one side, the other side, while not correctly called a channel program, performs the same function.

Modifying MMDF configuration

When you perform the initial MMDF configuration, the configuration utility sets up the appropriate *alias.**, **.dom*, **.chn*, and *mmdftailor* files. This initial configuration is sufficient for most sites; if your site configuration changes, you can rerun the configuration utility to update the MMDF files.

However, if this initial configuration does not take care of your site configuration needs, you must edit the MMDF configuration files manually after running the configuration utility. You might also need to edit your configuration files manually if your site configuration changes so that you can no longer configure MMDF with the configuration utility. See "Configuration utility limitations" (page 382) for more information.

Guidelines for manual configuration

Keep in mind the following guidelines when you modify your MMDF configuration files:

- Always perform configuration as *mmdf*.
- Always rebuild the hashed database with **dbmbuild**(ADM) after modifying files. See "Rebuilding the MMDF hashed database" (page 434) for more information.
- If you make any changes to the channel configuration (for example, to add or remove a channel), restart the **deliver**(ADM) daemon.

Changing the postmaster alias

Because RFC821/822 requires that every host provide the special postmaster mailbox, you should never change the **MSUPPORT** line in */usr/mmdf/mmdftailor* to anything other than *postmaster*. However, if the person responsible for mail administration at your site changes, you should change the *postmaster* alias in the */usr/mmdf/table/alias.ali* file.

For example, to change the *postmaster* alias to *hanna*, change the line in *alias.ali* to look like this:

```
postmaster:     hanna
```

NOTE If you do not define an alias (in the *alias.ali* file) that maps *postmaster* to a real user on the system, you must create a *postmaster* account (using the **sysadmsh**(ADM) command) to direct mail inquiries and problems.

You can designate anyone to receive undeliverable mail, but a local user is best because the address is simpler and, therefore, more likely to be a valid address. Also, with a local user, delivery of the message does not depend on a network connection that might be malfunctioning and therefore responsible for the original mail problem.

Maintaining user aliases

As you add new users to your system, you should update your */usr/mmdf/table/alias.user* file to map the new users to their home machines. For example, if you add *laurie* to the machine *poet* at your site, add the following line to *alias.user*:

```
laurie: laurie@poet
```

You can change the aliases in this file for other reasons, such as when you switch a user to another machine or the user leaves the company. For example, if a user leaves the company, you can redirect their mail to another person, such as their manager. To do this, place a line like the following in *alias.table*:

```
matt:  mark@scribe
```

If you create multiple-user aliases (mailing lists), you might need to update the information in your */usr/mmdf/table/alias.list* file periodically. For example, if you add a new alias called *docstyle*, add the following lines to *alias.list*:

```
docstyle:              docstyle-outbound@list-processor
docstyle-outbound:     joan,kelly,hanna,laurie,tammy,meg,teresa
docstyle-request:      joan
```

Changing the host name

When you install your operating system, the installation procedure prompts you to provide a name for your machine. This machine name is also called the HOST NAME or SYSTEM NAME.

If you decide to change your host name after installing the operating system, use the following procedure:

1. Log in as *root*.
2. Change the host name using the **uname**(C) utility. Use the following command, where ***new_name*** is the new name of your host:

   uname -S *new_name*

   For example, to change a machine name to *elgrande*, enter the following command:

   uname -S elgrande

   For more information on the **uname** command, see the **uname**(C) manual page.
3. Now, log in as *mmdf* and verify that you are in the */usr/mmdf* directory.
4. Edit the *mmdftailor* file and change all occurrences of the old host name to ***new_name***.
5. Change to the */usr/mmdf/table* directory and edit any **.dom* and **.chn* files, changing all instances of the old host name to ***new_name***.
6. Now, verify that you are in the */usr/mmdf/table* directory and rebuild the hashed database using the following command:

   ./dbmbuild

7. Now, log in as *root*, and bring the system down to single-user (system maintenance) mode with the following command:

   init 1

8. At the prompt, enter ⟨Ctrl⟩**d** to bring the system back up to multiuser mode.
9. If you are routing mail over TCP/IP, verify that the */etc/hosts* file contains the new host name.

Now, MMDF uses the new host name.

Hiding your host name

If, after running the initial MMDF configuration with the configuration utility, you register your top-level domain or subdomain with the NIC (Network Information Center), you might decide to hide the host name behind the domain name. In this case, people outside your organization can send mail to people at your site without knowing the host name where they receive their mail.

To hide your host name, you must edit the */usr/mmdf/mmdftailor* file and change the **MLDOMAIN**, **MLNAME**, and **MLOCMACHINE** keywords. For example, if you are not currently hiding the host name, these lines look like the following:

```
MLDOMAIN        npr.COM
MLNAME          scribe
```

To change the configuration to hide *scribe* behind the *npr* domain name, add **MLOCMACHINE** and change these parameters so they look like this:

```
MLDOMAIN  com
MLNAME npr
MLOCMACHINE scribe
```

Now, people can send mail to *andrei@scribe.npr.com* using the *andrei@npr.com* address (they no longer have to specify the host name, *scribe*).

Changing your smart host

If your configuration changes so that you no longer rely on a smart host for complete information about the network, or your smart host changes, you must modify the **MCHN** lines for **badhosts** and **badusers** in *mmdftailor*. These lines look something like the following:

```
MCHN badhosts, show="Last-Chance Routing", pgm=smtp, tbl=smtpchn,
     ap=822, host="rocky.npr.COM", confstr=scribe.npr.COM
MCHN badusers, show="Last-Chance Routing", pgm=smtp, tbl=smtpchn,
     ap=822, host="moose.npr.COM", confstr=scribe.npr.COM
```

In this case, change the value of the **host** parameter (in this example, *rocky.npr.COM* and *moose.npr.COM*) in each **MCHN** definition to the name of the new smart host. For more information about smart hosts, see "Hide your host name?" (page 388).

Changing error logging levels

By default, the MMDF configuration script sets logging levels with the **MMSGLOG** and **MCHANLOG** keywords in the *mmdftailor* file. These keywords are set to the FAT logging level (to log fatal errors only) and look like the following:

```
MMSGLOG         level=FAT
MCHANLOG        level=FAT
```

You can change these logging levels to a higher level, such as FST. In addition, you can use the **AUTHLOG** keyword to control authorization information. For a complete list of available logging levels, see the **MCHANLOG** section in **mmdftailor**(F) manual page.

Converting from XENIX

If you are converting from a XENIX to UNIX system or you have a mixed UNIX/XENIX system network, you must convert your XENIX aliases file and your UUCP routing files (as appropriate). (If you install your UUCP configuration files before running the MMDF configuration utility, you only have to convert the alias files.) This section explains how to convert these files.

Before converting your XENIX files, you must install and configure UUCP. You must install and configure these packages so that the files that you use to create MMDF alias, channel, and domain files contain the correct information.

If you did not install the entire distribution, you should install the UUCP and MAIL packages using the **custom**(ADM) utility. See your *Installation Guide* for information on installing packages with **custom**. For information on configuring UUCP, see the "Building a remote network with UUCP" chapter of this guide.

MMDF provides the utilities listed in Table 14-7 in the */usr/mmdf/table/tools* directory for converting files from XENIX to MMDF.

Table 14-7 MMDF conversion utilities

Utility	Converts this XENIX file:	To these MMDF files:
mmdfalias	/usr/lib/mail/aliases	/usr/mmdf/table/alias.list /usr/mmdf/table/alias.user
uulist	/usr/lib/uucp/Systems	/usr/mmdf/table/uucp.chn /usr/mmdf/table/uucp.dom

For more information, see the **mmdfalias**(ADM) and **uulist**(ADM) manual pages. Note that **uulist** uses **uuname**(C) to retrieve the list of accessible sites.

Use the following procedure to convert the XENIX files:

1. First, use the following command to make sure *aliases* and *top* are readable by *mmdf*:

   ls -l /usr/lib/mail/aliases /usr/lib/mail/top

   If the *mmdf* user does not have permission to read these files, use **chmod**(C) to change the permissions. Log in as *root* and enter the following command:

   chmod +r /usr/lib/mail/aliases /usr/lib/mail/top

2. Now, log in as *mmdf* and enter the following command:

   cd /usr/mmdf/table

3. Use the following commands to convert the routing or alias files, as appropriate:

   - To convert alias files, enter:

     tools/mmdfalias

     The **mmdfalias** utility creates the files *alias.list* and *alias.user* in the */usr/mmdf/table* directory.

   - To convert UUCP routing files, enter:

     tools/uulist

     The **uulist** utility creates the files *uucp.chn* and *uucp.dom* in the */usr/mmdf/table* directory.

   If any of the MMDF output files already exist, the MMDF utility backs up the original ***filename*** to ***filename-***.

4. Now, make sure that the permissions on each of the output files are set to 644, and that the owner and group are *mmdf*. Use the following commands:

   cd /usr/mmdf/table
   ls -l alias.list alias.user uucp.chn uucp.dom

   The permissions should look something like the following:

```
-rw-r--r--   1 mmdf     mmdf         150 Jun 11 1992 alias.list
-rw-r--r--   1 mmdf     mmdf         120 Jun 11 1992 alias.user
-rw-r--r--   1 mmdf     mmdf          86 Apr 17 14:23 uucp.chn
-rw-r--r--   1 mmdf     mmdf          61 Apr 17 14:23 uucp.dom
```

   If the permissions for any of these files are wrong, use **chmod**(C) to modify them.

Now, all the XENIX routing and alias files are converted to MMDF.

If you are converting to MMDF from XENIX or another UNIX system, the system mailboxes for the users on the system are in the XENIX-style (older UNIX system) format. XENIX format uses the "From<space>" lines to delimit between messages in mail folders; MMDF uses ⟨Ctrl⟩**a** characters. You should convert the system mailboxes to use the new MMDF format using the **cnvtmbox**(ADM) utility.

Use the following procedure to do this:

1. First, verify that users on the system are not accessing the system mailboxes.
2. Log in as *root*. (You must have read permission on the mailbox that you are converting.)
3. Now, change to the system mail spool directory (generally, */usr/spool/mail*).
4. For each system mailbox in the mail directory, run the following commands:

   /usr/mmdf/bin/cnvtmbox *old_mailbox new_mailbox*
   mv *new_mailbox old_mailbox*

By default, the **mailx**(C) program maintains the current format of every mail folder that users read with **mailx**. For example, if a mail folder is in MMDF format (with ⟨Ctrl⟩**a** delimiters), **mailx** uses this format when adding messages to the folder. If a user runs **mailx** with a folder that has not been converted to MMDF format, it prompts the user to convert the folder.

If any users on the system use an MUA other than **mailx**, advise these users to run **cnvtmbox** on each of their mail folders. As long as users have read permission on a mail folder and write permission on the output, they can convert it to the MMDF format with **cnvtmbox**.

Modifying UUCP host configuration

If you change your UUCP host configuration, use the information in this section to update your MMDF configuration files.

Specifying options to uux

If you use UUCP to route mail, you can specify the options that **uux**(C) should invoke when sending files and executing commands on the remote machine. By default, MMDF uses the following options to **uux**:

```
uux - -r
```

This setting specifies that UUCP queue the job until the next time **uucico**(ADM) runs rather than transfer it immediately (see **uucico**(ADM) for more information).

If you want to change this, add the **UUXSTR** keyword to the */usr/mmdf/mmdftailor* file. For example, if you want **uucico** to run immediately, the following **UUXSTR** setting is correct:

```
UUXSTR "uux -"
```

In general, we recommend that you use one of the following **UUXSTR** settings:

```
UUXSTR "uux -"
UUXSTR "uux - -r"
UUXSTR "uux - -r -gA"
```

For more information on these options, see the **uux**(C) manual page.

Adding or removing a UUCP host

To add a host that you access via UUCP to your system, use the following procedure:

1. Add the UUCP path to route mail to the host to the */usr/mmdf/table/uucp.chn* file. For example, to add the UUCP host *dudley*, add the following line to *uucp.chn* on the local host:

   dudley.UUCP: dudley!%s

   The "%s" specifies to use the remainder of the address indicated in the "To:" mail header from this point.

2. Add the UUCP host name to the *uucp.dom* file. For example:

   dudley: dudley.UUCP

3. Now, rebuild the hashed database with **dbmbuild**.

To remove a UUCP host, remove the entries for that host from the *uucp.chn* and *uucp.dom* files and rebuild the hashed database.

Adding or removing an SMTP host

Use the following procedure to add an SMTP host to your MMDF configuration:

1. For each new host in */etc/hosts*, add the fully qualified host name and IP address to the */usr/mmdf/table/smtp.chn* file. For example, to add two new hosts *chocolate* and *truffle*, enter:

   chocolate.npr.COM: **123.456.789.4**
   truffle.npr.COM: **123.456.789.5**

2. Now, edit the */usr/mmdf/table/smtp.dom* and match the unqualified SMTP host name to the fully qualified host name. For example, enter:

   chocolate: **chocolate.npr.COM**
   truffle: **truffle.npr.COM**

3. Now, use **dbmbuild**(ADM) to rebuild the MMDF hashed database.

Use the information in "Testing MMDF configuration" (page 434) to check the new SMTP configuration.

Running multiple deliver daemons

If you configure MMDF to route mail on channels in addition to the local channel, you must restart the **deliver** daemon and modify the */etc/rc2.d/S86mmdf* system startup file to tell **deliver** about the new channels.[5] The **deliver** daemon periodically checks each channel's queue for mail to deliver. This is known as "sweeping the queues".

"Modifying MMDF system startup" (page 402) explains how to start one **deliver** process for all the channels that you have configured. When you have a single **deliver** program managing a number of channels, **deliver** goes through the channels individually and tries to deliver all the messages in a channel's queue before going on to the next channel. You can configure your system to start multiple **deliver** programs, each servicing a single channel, instead. In this case, the **deliver** daemons work in parallel. This is a desirable configuration for a mail gateway machine because it increases the overall mail bandwidth of the machine.

NOTE For each **deliver** process that you initiate, system overhead increases.

5. See "Modifying MMDF system startup" (page 402) for information on how to do this.

To start separate **deliver** processes to manage each channel, use the following information to add **deliver** startup lines for each channel to the */etc/rc2.d/S86mmdf* file.

If you removed the **-clocal** from the **deliver** startup line in your *S86mmdf* file, the **deliver** startup line looks like the following:

```
/bin/su mmdf -c "/usr/mmdf/bin/deliver -b"
```

To create separate **deliver** processes, use the **-c** option to add lines for each channel. The following example shows separate **deliver** processes for each of the local, UUCP, and SMTP channels:

```
/bin/su mmdf -c "/usr/mmdf/bin/deliver -b -clocal"
/bin/su mmdf -c "/usr/mmdf/bin/deliver -b -cuucp"
/bin/su mmdf -c "/usr/mmdf/bin/deliver -b -csmtp"
```

If you have a lot of mail traffic on a single channel, you can start more than one **deliver** process running on that channel. To do this, simply add more than one **deliver** startup line for the channel in */etc/rc2.d/S86mmdf*. For example, to start three **deliver** processes to sweep the queue for the SMTP channel, add these lines:

```
/bin/su mmdf -c "/usr/mmdf/bin/deliver -b -csmtp"
/bin/su mmdf -c "/usr/mmdf/bin/deliver -b -csmtp"
/bin/su mmdf -c "/usr/mmdf/bin/deliver -b -csmtp"
```

Keep in mind that each **deliver** process that you start increases the load on the system.

Specifying different time intervals for deliver

The **MSLEEP** parameter in the *mmdftailor* file determines the default time interval for the **deliver** daemon to sweep through the queues for all the channels. The default setting for **MSLEEP** is 600, meaning that **deliver** sweeps the queues every 600 seconds, or 10 minutes. We recommend that you set up **deliver** to run every 60 seconds. To use this interval for all the channels, change the value of **MSLEEP** in *mmdftailor* to 60.

If you create a separate **deliver** daemon for each channel, you can set up a different time interval for each **deliver** process. To do this, add the **-T** option to the appropriate **deliver** startup line in */etc/rc2.d/S86mmdf*. For example, to run **deliver** on the local channel every 30 seconds and on the UUCP channel every 5 minutes, put the following lines in *S86mmdf*:

```
/bin/su mmdf -c "/usr/mmdf/bin/deliver -b -clocal -T30"
/bin/su mmdf -c "/usr/mmdf/bin/deliver -b -cuucp -T300"
```

Rebuilding the MMDF hashed database

The hashed database gives MMDF quick access to alias and routing information. You must rebuild this database whenever you change *mmdftailor* or any of the alias, channel or domain tables.

To rebuild the database, log in as *mmdf* and run the following commands:

```
cd /usr/mmdf/table
./dbmbuild
```

If any of the files in */usr/mmdf/table* are missing when you rebuild the database, **dbmbuild** reports that they are missing. See the **dbmbuild**(ADM) manual page for full details.

Testing MMDF configuration

Once you've modified your MMDF configuration and rebuilt the hashed database, use the information in this section to test the new configuration.

Checking for MMDF problems

Use the **checkup**(ADM) command to examine the full MMDF system on the local machine and report any inconsistencies. Normally, **checkup** reports all problems, including correct states; **checkup** marks problems with two asterisk characters (**) and and encloses advisory information in square brackets. You can tell **checkup** to report only the problems using the **-p** option. For example, enter the following command:

```
/usr/mmdf/bin/checkup | more
```

The following example shows the output from **checkup**:

```
**  Asterisks indicate potentially serious anomalies.

Tailor file             : /usr/mmdf/mmdftailor
**    Unknown tailor    : 'MLCKTYPE advisor'
```

In this case, **checkup** does not recognize the **MLCKTYPE** parameter **advisor** because it is misspelled (the correct parameter name is **advisory**).

If the permissions on any of the MMDF configuration files are incorrect, **checkup** lists this information.

Testing mail addresses

Use the **checkaddr**(ADM) command to test an individual address for validity. For example, to test the validity of the address, *andrei@scribe*, enter the following command:

/usr/mmdf/bin/checkaddr andrei@scribe

The **checkaddr** checks the address and displays the following message:

```
andrei@scribe: OK
```

This command is particularly useful for testing aliases after you have installed a new alias file. For more information, see the **checkaddr**(ADM) manual page.

Maintaining the MMDF system

Once you configure MMDF, the MMDF system requires minimal system administration. In addition to maintaining the alias files (as described in "Maintaining user aliases" (page 425)) and adding new machines to your network (see "Modifying MMDF configuration" (page 424)) you must perform the following tasks:

- Monitor space in the */usr/spool/mail* directory where mail is received, and in the */usr/spool/mmdf* directory where queued mail is stored.
- Clean outdated mail from the mail queues.
- Remove log files periodically from the */usr/mmdf/log* directory.

The following sections describe the utilities you can use to maintain your MMDF system.

Checking the status of mail queues

Use the **checkque**(ADM) program to check the status of the mail queues; this utility reports on the number of messages waiting for delivery. For example, enter the following command:

/usr/mmdf/bin/checkque

Here is an example of output from **checkque**:

```
Thu May  9 16:07:  2 queued msgs / 80 byte queue directory
                   4 Kbytes in msg dir

  2 msgs    4 Kb (local   ) local       :  Local Delivery
                   deliver start         :  Thu May  9 15:15
                   deliver message       :  Thu May  9 15:15
                   deliver end           :  Thu May  9 15:15 / 0 hours
  *** WAITING **   First message         :  Wed Apr 24 16:03
```

If a queue is backed up with waiting mail, you can manually force **deliver**(ADM) to deliver the mail using a command like the following:

deliver -w -clist,uucp

The **-c** option specifies the channels to be processed and **-w** forces **deliver** and the channel programs to output informative messages as they try to deliver the mail. You can review the output for abnormalities, such as a rejected sender or recipient. For a complete description of **checkque**, see the **checkque**(ADM) manual page.

Removing old mail from the queues

Use the **cleanque**(ADM) program to remove outdated files from the mail queues. Use **crontab**(C) as *mmdf* to run **cleanque** daily. You might want to run **cleanque** more often, depending on your mail volume. You can also run **cleanque** using **cron**(C). To do this, create a *crontabs* file for *mmdf* in the */usr/spool/cron/crontabs* directory that includes the following line:

```
0    0    *    *    0-6    /usr/mmdf/bin/cleanque
```

You can also run **cleanque** by hand whenever you suspect a problem with mail delivery. The **cleanque** manual page provides a complete description of this program.

Monitoring the size of log files

You should periodically check the size of log files in */usr/mmdf/log*. To limit the amount of log data that accumulates in a log file, move it to another filename, for example *chan.log-*.

If you move aside a log file, make sure that *mmdf* owns the new empty file. To back up the log file and create a new empty file, enter the following commands as *mmdf*:

cd /usr/mmdf/log
cp chan.log chan.log-
cat /dev/null > chan.log

Advanced MMDF configuration tasks

This section describes some additional configuration tasks that you can use to customize your MMDF system. In general, most sites do not need to perform any of the configuration procedures described in this section.

Changing the location of system mailboxes

By default, MMDF delivers mail to a file named with the user's login in the */usr/spool/mail* directory. If you want MMDF to deliver mail to a file or directory other than the default file, you can add lines (anywhere in the *mmdftailor* file) using the following **MDLVRDIR**, **MMBXNAME**, and **MMBXPROT** keywords.

Only one of **MDLVRDIR** and **MMBXNAME** can be non-null. If **MDLVRDIR** is null, MMDF delivers to the user's home directory; if **MMBXNAME** is null, MMDF uses the user's login as the name of the mailbox file. You can also change the location of the spool directory by setting **MDLVRDIR** to be non-null.

MMBXPROT sets the protection mode on mailbox files using the same set of octal numbers that you use to change access permissions on files with **chmod**(C). For example:

```
MDLVRDIR ""
MMBXNAME ".mailbox"
MMBXPROT 0600
```

In this example, MMDF delivers to a *.mailbox* file in the user's home directory and sets the file protection so that only the owner can read or write to the file.

Specifying the MMDF "signature"

You can change the signature that MMDF uses when notifying senders of mail delivery problems by setting the **MSIG** keyword in the */usr/mmdf/mmdftailor* file. This message should indicate which mail routing system was responsible. For example:

```
MSIG      "MMDF Mail System"
```

With this setting, when users receive "Failed" mail messages from MMDF, the message appears to be from "MMDF Mail System".

Configuring MMDF authorization

The MMDF authorization feature allows you to control the flow of mail through your host. For example, if your site is on the Internet (you registered your domain name with SRI), you might want to allow your employees (but no one else) to send mail from their home machines through your system to the Internet. In this case, you would want to set up MMDF to authorize mail from those users to pass through your machine, but prevent mail from other users from passing through.

With MMDF, you can control mail transferred through your host on a per-channel or per-user basis (or both). In other words, you can control the flow of mail on specific channels and from specific users.

To control authorization on a particular host, use the **auth** parameter to set an authorization level on the **MCHN** line in the */usr/mmdf/mmdftailor* file. MMDF provides the seven levels of authorization listed in Table 14-8.

Table 14-8 MMDF authorization levels

Level	Description
free	performs no authorization checks (default)
inlog	logs incoming authorized and unauthorized access, but allows mail to pass
outlog	logs outgoing authorized and unauthorized access, but allows mail to pass
inwarn	logs incoming unauthorized access, transfers the message, and sends a warning to the originator
outwarn	logs outgoing unauthorized access, transfers the message, and sends a warning to the originator
inblock	logs the incoming unauthorized access attempt and bounces the mail
outblock	logs the outgoing unauthorized access attempt and bounces the mail

The default channel authorization level is **free**; any channel that you do not set authorization for is set to **free**.

The following sections explain more about using these authorization levels to set up host- and user-based authorization.

Setting authorization for hosts

You can control the flow of mail both to and from your host and network by setting up host-based authorization controls. The most common reason that sites use host-based authorization is to make sure that hosts which are not part of a private network do not use that network to send mail to other parts of the world. For example, you can use host-based authorization to allow mail to enter your domain, but restrict the mail passing through it.

By default, MMDF performs all authorization on the next "hop" host in the address; if someone specifies a route through your host, MMDF on your host can authorize the next hop in that route. In other words, MMDF does not perform authorization based on the final destination unless the final destination is the next hop. However, you can perform authorization on the entire route

to the destination or from the source using the **dho** flag with the **auth** parameter to specify routing-based authorization controls. See "Routing-based authorization" (page 440) for more information.

You can control access to a particular channel based on whether the mail is inbound or outbound on the channel and whether the source or the destination of the mail is authorized. To do this, use one (or more) of the four tables listed in Table 14-9.

Table 14-9 Host-based authorization tables

Table	Description
insrc	When mail arrives from a particular host (the source host), this table verifies that that host is allowed to send mail to the destination host by authorizing either the destination host or the channels used to access the destination host.
outsrc	When mail arrives from the source host via a particular channel, this table determines if that source host or channel is allowed to send mail to the destination host.
indest	When mail is sent to a particular host or via a particular channel, this table determines whether the source host is allowed to send mail to the destination host or via that channel.
outdest	When mail is sent to a particular host, this table verifies that the source host or channel is allowed to send mail to the destination host.

For example, you might want to set up MMDF so that only hosts that belong to a certain network can send mail out on a specific channel. This could be the case if your site has limited resources and you do not want to allow other sites to pass mail through your site.

Use the following procedure to set up the host-based authorization controls:

1. First, log in as *mmdf*. (You must perform all MMDF configuration as *mmdf*.)
2. Use the **MTBL** keyword to define a table in */usr/mmdf/mmdftailor*. For example:

   MTBL "world-auth", file="authinfo/world"

   The *authinfo/world* table specifies the privileges for all the hosts that do not belong to the network. This table is stored in the file *world* in the directory */usr/mmdf/table/authinfo*; you can specify locations like this for all the tables.

3. Now, use **MCHN** to define two channels in *mmdftailor*: one channel for the hosts that belong to the network, one for the hosts that do not belong.

   For example, define a channel for the *sconet* network:

   MCHN sconet, auth=free, show="SCONET Delivery",
   ap=822, mod=imm

   (The "auth=free" authorization setting is the default; you do not have to specify it explicitly as in this example.)

   Now, define a channel for all the hosts (**world**) not in the *sconet* network (this appears as one line in *mmdftailor*):

   MCHN world, auth=inblock, indest="world-auth",
   show="WORLD Delivery", ap=822, mod=imm

   In this case, anyone can send mail out on the **world** channel, but MMDF checks the *world-auth* table to authorize the destination of mail arriving on this channel.

4. Now, create the channel tables (**.chn* files in the */usr/mmdf/table* directory) for the channels that you define in *mmdftailor* and include descriptions of each host accessed via that channel. (See "Channel files" (page 418) for more information.)

5. Finally, create the *world-auth* table and include lines like the following:

   ```
   world:
   local:
   sconet: moocow.uucp
   ```

   The entries on the LHS of the authorization table specify that if the destination for a message is on either the **world** or **local** channels, MMDF authorizes anyone using the **world** channel as an input channel to send mail.

   The entry on the RHS of the **sconet** channel entry specifies the hosts and channels that are authorized to send outgoing mail using the **sconet** channel. In this case, *moocow.uucp* is the only machine allowed to pass mail into the *sconet* network.

6. Now, rebuild the hashed database using the information in "Rebuilding the MMDF hashed database" (page 403).

Routing-based authorization

You might want to set up authorization to keep mail that enters your domain from leaving your domain. In other words, people can send mail from inside the domain to people outside the domain as well as to other people in the domain, but people outside the domain cannot send mail through your domain to another destination.

For example, if your domain includes machines in different cities with links to different outside computers, people from outside the domain might use these links to send mail from one city to another, through your domain. In this case, you can set up MMDF authorization to prevent people from using your system to transfer mail.

Because MMDF performs authorization only on the next "hop" in the address, you cannot use the procedure described in the previous section to set up authorization control. To do this, you must use the **auth** parameter with the **dho** flag to specify routing-based authorization controls.

The following procedure explains how to set up routing-based authorization for the hosts that are not in the *nprnet* domain:

1. Log in as *mmdf*.
2. Define a table in */usr/mmdf/mmdftailor*. For example:

   MTBL "world-auth", file="authinfo/world"

   This defines the *authinfo/world* file that contains the authorization information for the **world** channel.
3. Now, specify a channel for your domain. For example, if the domain name is *npr.com*, create an **MCHN** entry like the following:

   MCHN nprnet, auth=free, show="NPRNET Delivery",
   ap=822, mod=imm
4. Now, define a channel for the rest of the hosts that are not in the local domain (again, this appears as one line in *mmdftailor*):

   MCHN world, auth=inblock, auth=dho, indest="world-auth",
   show="WORLD Delivery", ap=822, mod=imm

   The "auth=indest" parameter specifies that when **world** is the input channel, MMDF checks the *authinfo/world* file to verify that the inbound host is authorized to send mail to the destination.

   When you specify the "auth=dho" parameter on a channel, MMDF replaces the "host" (in host-based authorization) used to check authorization with a route. The route is either from the source or to the destination, depending on which flag (from Table 14-9) that you specify. MMDF replaces the local section of the route (the user's name) with the string "username". Then, MMDF compares this route to the route specified in the message, to determine if the message is authorized or not.
5. Create the channel tables in the */usr/mmdf/table* directory for the channels that you define in *mmdftailor*. (See "Channel files" (page 418) for more information.)

6. Finally, create the *authinfo/world*, and include entries like the following:

```
world:
username@npr.com:
username@larry.npr.com:
username@moe.npr.com:
username@curly.npr.com:
```

   This table authorizes MMDF to deliver any mail addressed to people in the *npr.com* domain arriving or leaving on the **world** channel. This does not allow mail to pass through the **nprnet** channel to a destination outside the *npr.com* domain.

7. Now, rebuild the hashed database with **dbmbuild**.

Setting authorization for users

To configure MMDF authorization on a per-user basis, you must first set up the authorization level on the channels that you want to restrict as described in the previous section. Once you set up channel authorization levels, you must create a table in the */usr/mmdf/table* directory that maps user names to authorization levels and then declare this table in the */usr/mmdf/mmdftailor* file.

The following is the format for setting up the per-user authorization table:

username: ***keyword channel*** [, ***channel***]

The ***username*** can be either a local or remote user name, ***keyword*** is one of the keywords for the actions that you can authorize users to perform, and ***channel*** is the channel name from the **MCHN** line.

Table 14-10 lists the action authorization keywords:

Table 14-10 MMDF authorization keywords

Keyword	Description
both	allows user to send and receive mail
send	allows user to send mail only
recv	allows user to receive mail only
expire	expires access privileges for the user (and includes this information in any error mail)

MMDF treats any action not listed in the table above as **expire**, except that MMDF sends the text of the action instead of "expire" to the user in error mail.

Use these steps to set up authorizations for specific users:

1. First, log in as *mmdf*.
2. Use the information in "Setting authorization for hosts" (page 438) to set up authorization for any channels that you want to restrict.

3. Declare the user authorization table in the */usr/mmdf/mmdftailor* file. For example, if the name of the user authorization table is *auth.user*, the **MTBL** declaration looks like this:

   ```
   MTBL auth, file="auth.user", show "Per-user authorization"
   ```

   Note that you must call the per-user authorization table "auth"; MMDF treats any table called "auth" as the per-user authorization table.

4. Now, create the user authorization table in the */usr/mmdf/table* directory. Edit the file and include one line for each user to whom you want to grant mail access. Any users not listed in the user authorization table are not authorized to use any channel except the ones set to the **free**, **inlog**, or **outlog** authorization levels.

   In our example, MMDF allows users who are not listed in the user authorization table to pass mail in and out through the UUCP channel and in through the TCP/IP channel.

5. To set up access authorizations for a local user, specify the unqualified user name. In the following example, the local user *andrei* can both send and receive mail on the TCP/IP and UUCP channels:

   ```
   andrei: both smtp,uucp
   ```

   However, if you set up host-based authorization to restrict access to one of these channels, for example UUCP, *andrei* might not be authorized to send or receive mail on that channel.

   Because mail on the local channel is not restricted, *andrei* can pass mail through this channel even though the user authorization list does not include "local" in the list of channels.

6. To set up access authorization for a remote user, specify the fully qualified address of that user. For example, to allow *natasha* on the machine *kgb.GOV* to send mail through TCP/IP on this host, add a line like the following to the authorization table:

   ```
   natasha@kgb.GOV: send smtp
   ```

   If mail arrives for *natasha* through UUCP, or if she tries to send mail through the UUCP channel, MMDF rejects the mail.

   If you want to expire a particular user's access and tell MMDF to send an error message, add the message to the user authorization table line for that user. For example, to expire *rocky@squirrel.COM*'s access and send the text "No more mail for you!", include a line like this one:

   ```
   rocky@squirrel.COM: "No more mail for you!" uucp
   ```

7. Rebuild the hashed database with **dbmbuild**.

Setting host and user authorization controls

The default authorization algorithm allows MMDF to deliver a message if either the user-based or host-based authorization tables allow it. Thus, if a user table authorizes a user to send mail over a channel, but the host table does not, that user can still use the channel (and vice versa).

If you want to require that both the user and host tables authorize access to a channel, use the **hau** keyword with the **auth** parameter in the **MCHN** channel declaration in *mmdftailor*.

Understanding authorization log files

If you set the **level** parameter to the **AUTHLOG** keyword to an authorization level of FST (to give full statistics) or higher[6], MMDF maintains a complete log of authorization attempts and reasons for failure or success.

For example, to set the authorization logging level to save all diagnostic messages to the */usr/mmdf/log/mmdfauth.log* file, set **AUTHLOG** in the *mmdftailor* file like this:

AUTHLOG /usr/mmdf/log/mmdfauth.log level=FTR

The format of authorization messages in the log file is similar to other MMDF log files. Each message includes the date, time, message source, and message ID, followed by the log-specific information. For example:[7]

```
4/29  9:44:54 AU-0000:  msg.a000561: i='local' o='ucscc'
      a='lisa@rsre.AC.UK' r='CH' hi='' ho='username@rsre.ac.uk'
4/29  9:44:54 AU-0000:  msg.a000561: i='local' o='ucscc'
      a='jane@rsre.AC.UK' r='CH' hi='' ho='username@rsre.ac.uk'
4/29  9:44:55 AU-0000:  msg.a000561: END size='2102', sender='robert'
```

Table 14-11 describes the authorization message keys.

Table 14-11 Authorization keys

Key	Description
i	input channel
o	output channel
a	destination address
r	reason for authorization
hi	inbound host
ho	outbound host

6. See the section on **MCHANLOG** in the **mmdftailor**(F) manual page for a complete list of logging levels.
7. The authorization messages appear on one line; the examples in this section split the lines for readability.

In addition, the authorization message includes an "end of processing" message that describes the message sender and size. (In the example above, the third line is the "end of processing" message.)

Each authorization message can have either one or two reasons for authorizing a particular message. If the authorization line includes one reason, the "r" key specifies a single authorization code which describes both the inbound and the outbound authorization when you use host-based authorization. These codes are listed Table 14-12.

Table 14-12 Single-reason authorization codes

Code	Description
OH	outbound host/route
HC	outbound host/route and inbound channel
HH	inbound host/route and outbound host/route
CC	inbound channel and outbound channel
CH	inbound channel and outbound host/route
IH	inbound host/route

The following example uses the **CH** authorization code:

```
4/29  9:44:54 AU-0000:  msg.a000561: i='local' o='peaks'
      a='bob@rsre.AC.UK' r='CH' hi='' ho='username@rsre.ac.uk'
4/29  9:44:54 AU-0000:  msg.a000561: i='local' o='peaks'
      a='mike@rsre.AC.UK' r='CH' hi='' ho='username@rsre.ac.uk'
4/29  9:44:55 AU-0000:  msg.a000561: END size='2102', sender='cooper'
```

In this example, the authorized message has two recipients (*bob* and *mike*). The first authorization message shows that the inbound channel ("i") is the local channel and the outbound channel is **peaks**. The "a" key indicates that the recipient's address is *bob@rsre.AC.UK*.

The reason ("r") given for authorizing the message is **CH**; in other words, the inbound channel (**local**) has authorization to send mail to the given outbound host or route (specified by "ho"), in this case *username@rsre.ac.uk*.

The two-reason authorization codes describe the reason for authorization in terms of user-based authorization. Table 14-13 lists these codes.

Table 14-13 Two-reason authorization codes

Code	Description
IL	inbound channel, outbound = **LIST**
OL	outbound channel, dest = **LIST**
IS	inbound channel by sender
OS	outbound channel by sender
IR	inbound channel by receiver
OR	outbound channel by receiver
I	inbound channel, log unauthorized access*
O	outbound channel, log unauthorized access*

* MMDF only uses these authorization codes when you set "auth=inlog" or "auth=outlog".

The authorization message in the following example uses two-reason authorization (if no authorization is required for a channel, MMDF leaves the reason field ("r") empty):

```
4/29  9:53:09 AU-0000:  msg.a000653: i='local' o='npr'
      a='john@edxa.ac.uk' r='' r='OS'
4/29  9:53:10 AU-0000:  msg.a000653: END size='197', sender='david'
```

In this example, the message arrived (with no authorization required) on the local channel and is authorized to leave on the **npr** channel because the sender (*david*) is authorized to use it (**OS**).

Adding a new alias file

It is much easier to add entries to an existing alias file than to create one; see "Alias files" (page 412) and "Maintaining user aliases" (page 425) for more information. If you want to add a new alias file, use the following procedure:

1. Log in as *mmdf*.
2. Create a file in the */usr/mmdf/table* directory.
3. Verify that the new alias file is owned and group-owned by *mmdf* and set the permissions to 644.

   NOTE Do not give write permissions on alias files to other users. With write permissions to alias files, a user can forward another user's mail to another location. For this reason, you should designate a trusted user to make changes to the alias files as requested.

4. Add alias information to the new alias file. See "Alias files" (page 412) for information on the format of the various alias files.

5. Now, define the alias file in the */usr/mmdf/mmdftailor* file. Create an **MTBL** line to associate an abbreviated name and a more descriptive name with the new alias file. For example, the **MTBL** line in *mmdftailor* defines the alias file *alias.user* as:

   ```
   MTBL auser, file="alias.user", show="User Aliases"
   ```

   Now, you can refer to the file */usr/mmdf/table/alias.user* as "auser" throughout the rest of *mmdftailor*.

6. Add an **ALIAS** entry to *mmdftailor* to define information about the new alias file. For example, here is the **ALIAS** entry for *alias.user*:

   ```
   ALIAS table=auser
   ```

   The default parameters in the *mmdftailor* file are adequate for most requirements; however you can specify characteristics for the alias file using the following parameters:

   nobypass specifies that MMDF search the alias files first for all mail addresses on outgoing or incoming mail. In MMDF, placing a tilde (~) before an address causes the address to be interpreted as destined for its literal meaning. The **nobypass** parameter means that normal aliasing cannot be bypassed. For example, suppose the alias file contains this entry for George and **nobypass** is not set:

   ```
   george: george@vegan.edu
   ```

   If you enter:

   mail george

   then the mail goes to the *george* login on the *astoria* machine. If you then enter:

   mail "~george"

   then aliasing is disabled and mail is sent to a George on the local machine. If nobypass is set, entering the tilde character (~) has no effect and every mail request is translated by the alias files.

   public allows remote sites to expand aliases from this table using the SMTP **EXPN** command; you can also use **malias**(C) to display this information.

   trusted directs mail to be delivered to any file or process using the permissions of any user on the system (including *root*); only the super user should have access to modify a trusted alias file.

Note that MMDF searches the alias tables in the order that you list them in *mmdftailor*.

For more information, see the **mmdftailor**(F) manual page.

Adding a new domain file

Use the following procedure to create a new domain file:

1. Log in as *mmdf*.
2. Create a file in the */usr/mmdf/table* directory. Generally, you should name domain files for the domain that they describe. For example, the domain file for the *npr.COM* domain is called *npr.dom*.
3. Verify that the new domain file has the correct permissions (see step 3 in the previous section).
4. Add the information for the domain to the new domain file. See "Domain file format" (page 416) for more information.

   NOTE The entries on the left-hand side in this file must contain no leading whitespace.

5. Create an **MTBL** entry in *mmdftailor* for the new domain file. For example:

   ```
   MTBL nprdom, file="npr.dom", show="NPR Domain"
   ```

   See the **mmdftailor**(F) manual page for more information about additional parameters that you can use to specify additional properties about the table that you are defining with **MTBL**.
6. Add an **MDMN** entry for the new file:

   ```
   MDMN "npr.COM", show="NPR Domain", table=nprdom
   ```

   The first argument is the name of the domain.

Adding a third-party channel program

If you add a new channel program from an SCO add-on package, you must define and add a new channel file. A channel is the mechanism for delivering mail either to a mailbox on the local machine or across the network to a remote machine. The **MCHN** entries in *mmdftailor* define the channels available to MMDF for mail transport.

You can define additional channels for the network protocols configured on your system by specifying them in the *mmdftailor* file. Channel definitions look like this:

```
MCHN    uucp, show="UUCP Delivery", que=uucp,
        tbl=uuchn, ap=822, pgm=uucp, mod=imm
```

The first argument on the **MCHN** line is the name of the channel.

Use these steps to add a new channel:

1. Create an **MCHN** entry in */usr/mmdf/mmdftailor* and specify the parameters to use in your channel definition. For a complete list of the **MCHN** parameters, see the **mmdftailor**(F) manual page.
2. Create a channel file (**.chn*) in */usr/mmdf/table* and include entries for each host that you access via the new channel and how to route mail to those hosts. For more information, see "Channel files" (page 418).

Changing MMDF parameters

The MMDF parameters in the */usr/mmdf/mmdftailor* file allow you to redefine the certain MMDF variables. This section covers some of the more common MMDF parameters; for a complete list, see the **mmdftailor**(F) manual page.

AUTHLOG controls the authorization information. The following example shows the format of this parameter:

```
AUTHLOG /tmp/mmdf/mmdfauth.log, level=FST, size=40, stat=some
```

The **AUTHLOG** level must be at least FST, or MMDF does not save any authorization logging information. For more information, see the section on **MCHANLOG** in the **mmdftailor**(F) manual page.

MFAILTIME is the time (in hours) a message can remain in a queue before MMDF sends a failed mail message to the sender and purges the message from the queue.

```
MFAILTIME       168
```

MLCKTYPE allows you to specify the locking protocol for MMDF to use when locking users' mailboxes. This is useful if the users on the system use third-party MUA's that use a lock file that is different from the standard System V lock file. Set the **MLCKTYPE** parameter to one or more of the keywords in Table 14-14.

Table 14-14 MMDF locking keywords

Keyword	Lock file
advisory	System V fcntl( ) kernel locking protocols
v7	Version 7 and System V Release 3, and earlier locking protocols
xenix	XENIX locking protocols
all	all above locking protocols (default)

The default locking protocol is **all**; however the MMDF configuration utility sets **MLCKTYPE** to **advisory**.

With the **v7** keyword, MMDF creates a file called ***username.lock*** in */usr/spool/mail*. The **xenix** keyword specifies that MMDF create a file called */tmp/username.mlk*. In both cases, ***username*** is the name of the user's mailbox.

You can specify more than one locking protocol on the **MLCKTYPE** line. If you do, MMDF must successfully lock the user's mailbox using all the locking protocols before the mailbox is considered locked. For example, to use **advisory** and **xenix**, set **MLCKTYPE** like the following:

```
MLCKTYPE advisory, xenix
```

In this case, MMDF must lock the mailbox, using the **fcntl**() kernel file locking protocol *and* create a file called */usr/spool/mail/username.lock*. If it fails to perform both of these locks, MMDF must release the successful lock and try again later. Thus, if MMDF failed to perform the **fcntl** lock, it must unlink */usr/spool/mail/username.lock*.

MMAXHOPS specifies the maximum number of "Received:" or "Via:" lines a message can contain before the MMDF considers that the message is looping and rejects it.

```
MMAXHOPS        20
```

MWARNTIME specifies the time (in hours) that a message can remain in a queue before MMDF sends a warning message about delayed delivery to the sender.

```
MWARNTIME       72
```

MMSGLOG controls the logging information produced by the deliver **deliver**(ADM) and **submit**(ADM) programs. The following example shows the format of this parameter:

```
MMSGLOG /tmp/mmdf/mmdfmsg.log, level=FST, size=40, stat=some
```

For more information, see the **logs**(F) manual page and the section on **MCHANLOG** in the **mmdftailor**(F) manual page.

MCHANLOG controls MMDF logging, except for information controlled by **AUTHLOG** and **MMSGLOG**. See the **mmdftailor**(F) manual page for details.

MSLEEP determines the length of time (in seconds) that the **deliver** daemon sleeps between scanning the queues. By default, **MSLEEP** is set to 600 (10 minutes). We recommend that you set **MSLEEP** to 60 seconds:

```
MSLEEP    60
```

Getting more information

If you need more information to understand how the various components of MMDF interact, refer to the manual pages listed in Table 14-15.

Table 14-15 MMDF manual pages

Manual page	Description
checkaddr(ADM)	checks a mail address for validity
checkque(ADM)	reports on MMDF queue status
checkup(ADM)	checks MMDF system configuration
cleanque(ADM)	sends warnings and returns expired mail
cnvtmbox(ADM)	converts XENIX-style mailboxes to MMDF format
dbmbuild(ADM)	builds the MMDF database
dbmedit(ADM)	edits the MMDF database
deliver(ADM)	delivers mail to MMDF channels
list(ADM)	handles mailing lists
mmdf(ADM)	routes mail locally or over network
mmdfalias(ADM)	converts XENIX-style aliases file to MMDF
rmail(ADM)	submits remote mail received via MMDF
submit(ADM)	accepts mail to be handled by MMDF
uucico(ADM)	transfers UUCP files
uulist(ADM)	converts UUCP file to MMDF format
checkmail(C)	checks for submitted but not delivered mail
mail(C)	mails messages between users
uucp(C)	sends information between UNIX machines over serial lines
rcvalert(C)	alerts user when mail is received
rcvfile(C)	stores mail in a file automatically
rcvprint(C)	prints received mail automatically
rcvtrip(C)	informs sender that user is away
uname(C)	prints the name of the current system
uux(C)	transfers UUCP files and executes commands on remote system
logs(F)	logs MMDF messages
maildelivery(F)	describes the user delivery specification file
mmdftailor(F)	provides run-time tailoring
queue(F)	queues files for storing mail in transit
systems(F)	UUCP Systems file format

(Continued on next page)

Table 14-15 MMDF manual pages
(Continued)

Manual page	Description
tables(F)	describes alias, domain, and host tables
llog(S)	performs standardized information logging for programs
ml_send(S)	provides simple interface for mail submission
mmdf(S)	simplifies MMDF MUA mail submission and pickup
phs(S)	notes the MMDF transmission phase
tai(S)	gets site tailoring information

Chapter 15

Backing up filesystems

The main task of a system administrator is to ensure the continued integrity of information stored on the system. Files and filesystems can be damaged and data lost in the following ways:

- power interruptions (make certain you have a surge protector)
- hardware failures (particularly the hard disk)
- user errors (accidental removal of important files)

The importance of having up-to-date backups cannot be overstated. If your system has a number of active accounts, backups require daily attention. It is difficult to estimate the magnitude of a simple loss of data until an accident occurs and several weeks or months of work is gone in an instant.

A filesystem backup is a copy, on storage media (floppy disks or tape) of the files in the root filesystem and other regularly mounted filesystems (for example, the */u* filesystem). (See the Chapter 5, "Managing filesystems" in this guide for a discussion of filesystems.) A backup allows the system administrator (or user with the **backup** authorization) to save a copy of a filesystem as it was at a specific time.

This chapter explains how to use **sysadmsh**(ADM) to create backups of the root directory and other filesystems, and how to restore files from the backups.

You can also perform unattended backups; refer to "Unattended backups" (page 477) for more information.

(Another utility used for simple backups, **tar**(C), is discussed extensively in the "Using disks and tapes" chapter of the *User's Guide*. **tar** is not sufficiently sophisticated to perform scheduled backups; it is better suited to archiving groups of files.)

The tools discussed in this chapter present menus with simple options instead of the complex command lines used with the utilities **tar**(C), **cpio**(C), **backup**(ADM), and **restore**(ADM). The key to efficient backups is to save only what has changed from day to day, which (when used with **backup** and **restore**) normally requires extra bookkeeping.

Strategies for backups using sysadmsh

As system administrator, you should familiarize yourself with this chapter and create a backup schedule as instructed. When this schedule is complete, you have only to insert a media volume and respond to a series of prompts to perform your daily backups.

The primary purpose of the **sysadmsh** filesystem backup selection is to provide a dependable schedule of filesystem backups for systems with many users and large filesystems. The program automatically locates modified files and copies them to backup media. If your system has many users and a large number of files that are modified daily, the "scheduled" backup option uses a predefined schedule to make regular backups. When the Backups selection is invoked, the program presents each task as a menu option. To perform a task, simply choose the appropriate option from the menu and supply any required information.

For backups of an informal nature, **sysadmsh** includes an option for "unscheduled" backups. This allows the system administrator to perform a single, complete backup of a filesystem. (Note this type of backup covers the entire filesystem, not just modified files, and may require a number of storage media volumes.) If you intend to rely on unscheduled backups, be sure to perform one at least once a week.

Using the backup authorization

You must assign the **backup** authorization to a user to create or restore backups. (You must be *root* to restore an entire filesystem.) Ordinary users cannot make backups because they do not have access permissions for all files. The **backup** authorization allows you to assign a user the authority to create backups without giving them the *root* password.

Floppy drive backups and large systems

If your system has only a floppy drive, backups for large systems with several users can be time-consuming and use a great deal of media. A complete backup of a 20 Mbyte filesystem requires 15 1.2 Mbyte 96tpi diskettes, while a single 450-foot cartridge tape can store more than twice that amount. More importantly, diskettes require the presence of the operator to insert and remove floppies, whereas a single cartridge tape can be inserted and the operator need not remain by the system. If your system has a large number of users and just a floppy drive, you should install a cartridge tape drive, or make complete system backups once per week and warn your users to make individual backups of their own files on a regular basis.

Summary of utilities accessed

The **sysadmsh** accesses several utilities during the backup process. You do not need to be familiar with them. However, should you wish to use advanced options not discussed in this chapter, you will need to know how they are used and which reference pages to read. **sysadmsh** accesses the following utilities:

- **fsphoto**(ADM) is the main utility that controls the automated backup facilities.
- **fsave**(ADM) is the program that interacts with the user to perform the backup.
- **schedule**(ADM) is the backup database that specifies the media to be used, the filesystems to be backed up, and when to do so.
- **xbackup** and **xrestore**(ADM) are the XENIX backup utilities. These utilities are accessed when "xbackup" appears in the "Method" field of the Schedule table. (These utilities only work for XENIX filesystems.) **cpio** is the preferred method.
- **cpio**(C) is the default backup program. It is non-filesystem specific.

The **sysadmsh** Backups selection forms the "user-friendly" layer that isolates the user from the complicated syntax of these programs.

Preparations for scheduled backups

The only mandatory requirement for scheduled backups is the creation of a backup schedule. In addition, it is recommended that the system administrator follow the optional procedures for labeling, storing, and logging backups. A detailed explanation of backup levels is included at the end of this chapter in case you want to design a more complicated schedule.

Creating a backup schedule

The first step is to create a timetable for backups using the **schedule**(ADM) file. This file is located in the */usr/lib/sysadmin* directory and contains all the data needed for the system to perform a system backup, including:

- the name of your site or machine
- the media type and drive to be used
- a precise schedule of filesystems to be backed up

The sections that follow explain what changes should be made to the *schedule* file provided with your distribution.

Edit the schedule file

You can edit the *schedule* file with any text editor; make certain you are logged in as *root*. Example 15-1 shows the default *schedule* file. You can also use the following **sysadmsh** selection to edit the *schedule* file:

Backups ⇨ Schedule

sysadmsh uses the **vi**(C) editor by default, but you can set the **SA_EDITOR** environment variable to the editor you prefer. See **environ**(M) or **sh**(C) for an explanation of how to set environment variables. The subsections that follow explain the exact changes you need to make to this file.

Example 15-1 The schedule file

```
# SYSTEM BACKUP SCHEDULE
site machinename
# Media Entries
#
# 96 tpi 1.2 MB floppy 0
media /dev/rfd096ds15 k 1200 format /dev/rfd096ds15
# 96 tpi 1.2 MB floppy 1
# media /dev/rfd196ds15 k 1200 format /dev/rfd196ds15
# 135 tpi 1.44 MB floppy 0
# media /dev/rfd0135ds18 k 1440 format /dev/rfd0135ds18
# 135 tpi 1.44 MB floppy 1
# media /dev/rfd1135ds18 k 1440 format /dev/rfd135ds18
# Cartridge tape 1
# media /dev/rct0 k 60000 125000 150000 tape erase
# Mini cartridge drive (10MB)
# media /dev/rctmini k 8800 format /dev/rctmini
# Mini cartridge drive (20MB)
# media /dev/rctmini k 17200 format /dev/rctmini
# Mini cartridge drive (40MB)
# media /dev/rctmini k 37500 format /dev/rctmini
# 9-track tape drive
# media /dev/rmt0 d 1600 2400 1200 600

# Backup Descriptor Table
#   Backup   Vol.  Save for     Vitality        Label
#   level    size   how long     (importance)    marker
    0        -      "1 year"     critical        "a red sticker"
    1        -      "4 months"   necessary       "a yellow sticker"
    2        -      "3 weeks"    useful          "a blue sticker"
    3        -      "1 week"     precautionary   none

# Schedule Table
#               1 2 3 4 5   6 7 8 9 0   1 2 3 4 5   6 7 8 9 0
# Filesystem    M T W T F   M T W T F   M T W T F   M T W T F   Method
  /dev/rroot    0 3 3 3 3   2 3 3 3 3   1 3 3 3 3   2 3 3 3 3   cpio

# Alternate schedule for systems with /u filesystems
#/dev/rroot     0 x 3 x 3   2 x 3 x 3   1 x 3 x 3   2 x 3 x 3   cpio
#/dev/ru        3 0 3 3 3   3 2 3 3 3   3 1 3 3 3   3 2 3 3 3   cpio
```

Add the name of your site or machine

Simply change the ***machinename*** entry at the top of the file to the name you wish.

Select the media device that matches your configuration

Depending on your distribution media, the default drive is either 96tpi 1.2 Mbyte or 135tpi 1.4 Mbyte floppy drive 0, or Cartridge tape 1. The Cartridge tape 1 entry is reproduced in Example 15-2. The number signs (#) are comment symbols used to "comment out" text so that it is ignored by the program. Note that the default drive is the only one without a comment symbol. If you plan to use a drive other than the default, put a comment symbol in front of the default device and remove the comment symbol from in front of the drive you wish to use. The remaining drives should remain commented out.

NOTE Only one drive can be available at any one time.

Example 15-2 Default media entry

```
# Cartridge tape 1
media /dev/rct0 k 60000 125000 150000 tape erase
```

You can also customize or create a new entry to suit a device that is not listed. Example 15-2 includes the sizes of the 60, 125, and 150 Mbyte tapes. (The list of sizes is optional.) The size must be in Kbytes, thus 60 Mbytes is 60,000 Kbytes. For example, the following entry is appropriate for a 1.3 Gbyte DAT drive:

```
# 1.3 Gb DAT Device
media /dev/rct0 k 1300000 tape erase
```

NOTE Mini cartridge entries use sizes slightly smaller than the capacity listed on the cartridge because some storage is reserved for error correction code (ECC) data. For example, the 40 Mbyte mini cartridge drive entry uses 37500 rather than 40000.

Edit the backup descriptor table

Directly below the media drive lines is the Backup Descriptor table. This table, reproduced in Example 15-3, describes each backup level in terms of volume size, how long it is to be stored, how important it is, and how it is marked. The default entries should prove useful, but the volume size entries must be edited according to the type of media you are using.

Example 15-3 Backup descriptor table

```
#   Backup   Vol.  Save for     Vitality        Label
#   level    size   how long     (importance)    marker
    0        -      "1 year"     critical        "a red sticker"
    1        -      "4 months"   necessary       "a yellow sticker"
    2        -      "3 weeks"    useful          "a blue sticker"
    3        -      "1 week"     precautionary   none
```

If you are using floppy disks, leave the dashes in the "Vol. size" column as they are. This causes the backup program to take the volume size from the media entry for that device.

If you are using tapes or tape cartridges, replace each dash in the "Vol. size" column with the size (in kilobytes) of the tape volume. If you are using tapes that are all the same size for each backup level, replace each with the size of the tape you are using.

The last column contains label entries that are discussed in "Labeling your backups" later in this section.

Edit the backup schedule table

The default schedule assumes that backups are done every day. To make backups more efficient, they are broken into levels. Level 0 is the lowest level backup. A level 0 backup saves everything on the filesystem, while 1, 2, and 3 each back up only the files that have changed relative to the last lower-level backup. This concept is illustrated in Figure 15-1 with a stack of toy disks representing each level. Notice that the level 0 disk is the largest backup, and each of the others is progressively smaller. This is because each of the 1, 2, and 3 backups contain only the files that changed since the previous lower level backup. This figure also illustrates how these backups would be restored: first the level 0, followed by each of the latest 1, 2, and 3 that were done.

The concept of levels may seem needlessly complex at first, but consider what would happen on 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 only the files that changed most recently are saved, backups would be less time-consuming and, depending on the size of your media, consume fewer volumes. (See "A detailed explanation of backup levels" (page 474) for a more detailed discussion of backup levels.)

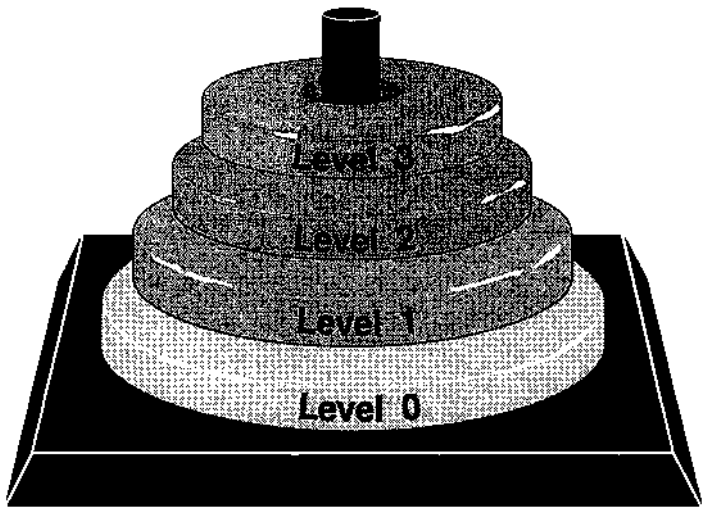


Figure 15-1 Backup levels

The example *schedule* file in this chapter includes an alternate schedule for a */u* filesystem that is commented out. Note that there is a backup done every other day for the root filesystem and once a day for the */u* filesystem. This is because the */u* filesystem (user accounts) changes much more frequently than the root filesystem, which contains the system files. An "x" means that a backup is not performed on that day for that filesystem.

If you do not have a */u* filesystem, then your user accounts are located in the root filesystem (in the directory */usr*). If this is so, the schedule table is preconfigured to back up the root filesystem. However, if you have added a */u* filesystem, you should edit the schedule table and remove the # symbols from the lines below the words "Alternate schedule for systems with /u filesystems" and insert a "#" symbol at the beginning of the original root filesystem entry. Make sure the table looks like Example 15-4. This ensures that backups are made of the additional filesystem.

Example 15-4 Backup schedule table

```
# Schedule Table
#               1 2 3 4 5  6 7 8 9 0  1 2 3 4 5  6 7 8 9 0
#Filesystem     M T W T F  M T W T F  M T W T F  M T W T F      Method
#/dev/rroot     0 3 3 3 3  2 3 3 3 3  1 3 3 3 3  2 3 3 3 3      cpio

# Alternate schedule for systems with /u filesystems
/dev/rroot      0 x 3 x 3  2 x 3 x 3  1 x 3 x 3  2 x 3 x 3      cpio
/dev/ru         3 0 3 3 3  3 2 3 3 3  3 1 3 3 3  3 2 3 3 3      cpio
```

Note that the Monday-Friday notation can be misleading; if a backup is postponed or unsuccessful (because of bad media, for example) then that same level backup is attempted again at the next scheduled backup. This offsets the schedule, but does not alter the established sequence of backups. The numbered scale of 1-0 above M-F is more accurate, but less useful to people, who work in day and week units.

In addition, if you add lines for other filesystems, you should take care not to schedule two level 0 backups of large filesystems on the same day; the process is lengthy and may slow your machine significantly.

Backup method field

The default backup format type is **cpio**. If you wish to use the **xbackup** format used with XENIX filesystems, replace the "cpio" with "xbackup". Note that the **xbackup**(ADM) utility only works on XENIX filesystems because it uses filesystem-specific information. The **cpio** format functions perfectly with UNIX, XENIX, AFS, and EAFS filesystems because it does not use such information. Use of the **xbackup** type is not recommended.

NOTE Remember that EAFS is the default filesystem type used by the operating system.

Labeling your backups

It is important to label your backup tapes with meaningful and accurate information. If your backups consist of a pile of haphazardly labeled tapes, it will be difficult to locate data at a later date.

Figure 15-2 is a suggested format for media labels.

Name of computer	Backup level Filesystem name Save until date	Date made
Name of backup person		Volume # of #

Figure 15-2 Sample media label

The date on the label, and the date from which you calculate the "save until" date, should be the date of the business day covered by the backup. This is to avoid confusion if it becomes necessary to restore information from this tape.

You may have noticed that the *schedule* file has a proposed color-coding scheme for easy reference, as emphasized in Example 15-5.

Example 15-5 Backup labeling scheme

```
#   Backup   Vol.  Save for     Vitality        Label
#   level    size   how long     (importance)    marker
    0        -      "1 year"     critical        "a red sticker"
    1        -      "4 months"   necessary       "a yellow sticker"
    2        -      "3 weeks"    useful          "a blue sticker"
    3        -      "1 week"     precautionary   none
```

If there is more than one tape for a single backup, mark the date label on each volume to indicate the volume number and number of volumes, such as "1 of 2" and "2 of 2" for a two-volume backup. Finally, place a label on the side of the box or enclosure marked with the name of the computer, the filesystem, and the backup level completed.

Keeping a log book

It is recommended that a written log book be maintained for each computer. In addition to maintenance information (such as when breakdowns occur and what was done about it), you should record the following information:

Date	Just as with the tape label, this date should be the last day covered by the backup.
Filesystem	This is the name of the device backed up on the current tape.
Backup level	This is the backup level of the current tape.
#Vols	This is the number of tape volumes.
Start/finish time	(Optional) The time from the start of a backup of a filesystem until the last error check is completed. The times are displayed after the backup is finished. The finish time will often be inaccurate, since you may be out of the room when the backup finishes, and the machine sits idle before you return.

If there are problems with the backup, record these in the log book as well, including any error messages that come to the screen.

Rotating backup media

Backup media should be used so that at least 6 to 12 months of media are left on file. The default *schedule* file includes a suggested rotation of 1 week on level 3 backups; 3 weeks on level 2; 4 months on level 1; and 1 year on level 0. This means that if you follow the default schedule, you can safely reuse your level 3 backups after 1 week, and so on.

Archiving backup media

All filesystems should be periodically backed up and archived offsite. In the event of a fire or natural catastrophe, the data can later be restored.

Performing a scheduled backup

This section describes how to perform a backup using a defined schedule. Do not attempt this until you have edited (or at least examined) the *schedule* file to make certain that it suits your needs.

The system administrator should schedule backups at times when few (if any) users are on the system. This ensures that the most recent version of each file is copied correctly.

A regular schedule of backups requires a good supply of media and adequate storage for them. Level 0 backups should be saved at least a year, longer if they are important. Lesser backups should be saved at least two weeks. Media volumes should be properly labeled with the date of the backup and the names of the files and directories contained in the backup. After a backup has expired, the media can be used to create new backups.

Using formatted media

If you use media that requires formatting, such as floppy disks or tape cartridges, you are advised to always have formatted volumes before you begin. The exact number of volumes depends on the number and size of files to be backed up. For details on how to format your media, see the "Using floppy disks and tape drives" chapter in this guide. You also have the option to do formatting from the **sysadmsh** program, but you cannot format media while a backup is in progress.

NOTE Formatting Irwin tape cartridges is not recommended. Best results are achieved with preformatted cartridges.

Starting the backup

To run your scheduled backup, follow these steps:

1. Invoke **sysadmsh** and select the following:
   Backups ⇨ Create ⇨ Scheduled

2. A menu is displayed that looks like the following:

```
Level 0 backup of filesystem /dev/rroot, 31 Aug
         tape size:         1200 Kb
         tape drive:        /dev/rfd096ds15
This tape will be saved for 1 year, and is critical.

M)ounted volume, P)ostpone, C)heck or F)ormat volumes, R)etension or H)elp:
```

   The media type displayed is the one entered in the *schedule* file. Load a tape or disk into the selected drive. Enter " m " to tell the program the volume is mounted, and press ⟨Return⟩.

3. The system displays the current date and the date of the last backup:

```
Level 0 backup of filesystem: /dev/rroot
Backing up all files
Generating list of pathnames for backing up ...
```

   This process takes a few minutes.

4. The system then begins to copy files to the drive. If a volume runs out of space, the program displays the following messages:

```
Reached end of medium on output
Insert volume 2 and press ⟨Return⟩ to continue or ''q'' to exit.
```

   NOTE If **xbackup**(ADM) is being used to make the backup, a slightly different prompt is displayed.

   Remove the present volume, insert a new volume, then press ⟨Return⟩. The program continues to copy files to the new volume. Repeat this step until the program displays the message:

```
Check critical volumes for format errors
```

5. When the backup is complete, the following menu is displayed:

```
M)ounted which volume, S)kip format check, or H)elp
```

   Level 0 backups should always be checked for format errors. Enter "m" to check your media. If you are checking the format, make certain you insert the first volume as instructed, or the backup aborts. If you do not want to check the volumes, enter "s".

6. If an error occurs, the backup is declared unsuccessful and is retried from the beginning. Your media could be bad, so replace it if errors persist. The menu keeps track of the volume being checked:

```
M)ounted which volume, E)rror on previous volume, D)one,
S)kip checks, or H)elp:
```

   When you are finished checking volumes, select "d".

After the backup is successfully performed, instructions are given on how to label the volumes. Make certain that you write-protect your volumes.

Performing an unscheduled backup

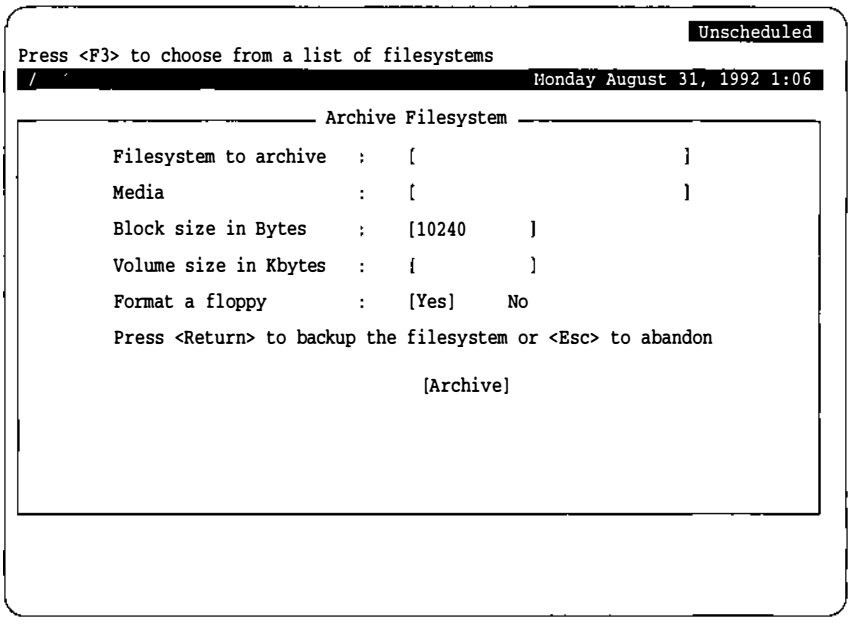
For backups of an informal nature, **sysadmsh** includes an option for "unscheduled" backups. This allows the system administrator to perform a single, complete backup of a filesystem without using a schedule. If your backup needs are simple, you can do an unscheduled backup on a regular basis. This type of backup covers the entire filesystem, not just modified files, and may require a number of storage media volumes. If you intend to rely on unscheduled backups, be sure to perform one at least once a week.

To create an unscheduled backup, follow these steps:

1. Invoke **sysadmsh** and select the following:

   Backups ➪ Create ➪ Unscheduled

2. The following menu is displayed:



3. Select the filesystem to back up by entering the name or pressing ⟨F3⟩ to get a point-and-pick list. The menu lists all filesystems found in the file */etc/default/filesys* (see **filesys**(F)). Use the arrow keys to select the filesystem you wish to back up and press ⟨Return⟩.
4. Next, select the media device to be used by entering the name or pressing ⟨F3⟩ to get a list. The block size is selected automatically.
5. Now enter the volume size in Kbytes. For example, specify a 60 Mbyte tape as "60000".

> **NOTE** Take care when selecting the number of the media device. For example, make certain that you do not select "Floppy Drive 1" (the secondary floppy drive) when you want "Floppy Drive 0" (the primary floppy drive). If you make this error, the backup is aborted and you must start over.

6. You can format as many volumes as you wish by inserting each volume one at a time into the drive and selecting Yes on the Format floppy. (Cartridge tapes can also be formatted, but this takes a great deal of time.)

7. Load a volume, tape or disk, into the selected drive, and press ⟨Return⟩. The system then begins to copy files to the drive, displaying the filenames as they are backed-up. If a volume runs out of space, the following is displayed:

```
Reached end of medium on output
Insert volume 2 and press ⟨Return⟩ to continue or ''q'' to exit.
```

8. Remove the first volume, insert a new volume, then press ⟨Return⟩. The program continues to copy files to the new volume. Repeat this step until the program displays the message:

```
DONE
```

If you are using floppies, you may need to repeat the last step several times before the backup is complete. You should label each volume as you remove it from the drive. For example, label the first volume "Volume 1", the second "Volume 2", and so on.

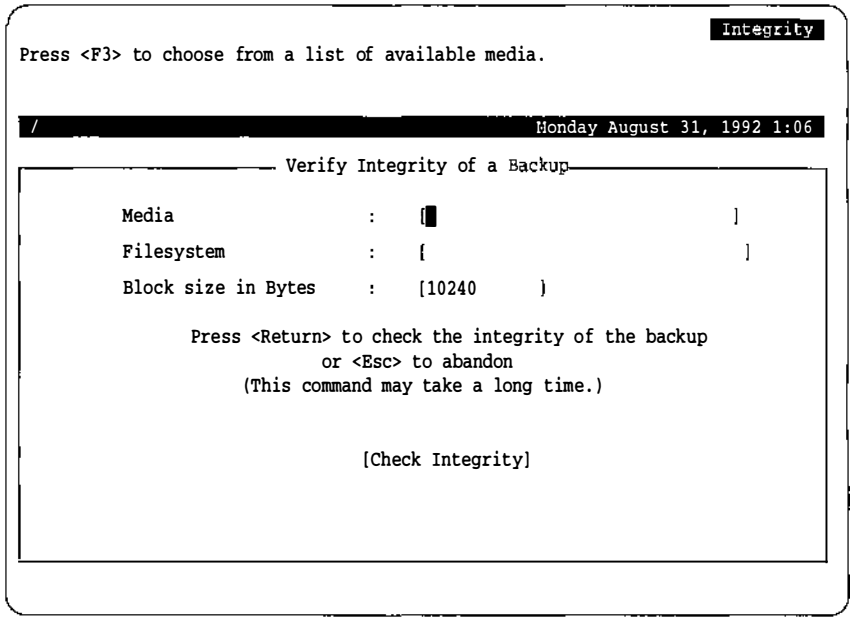
Verifying a backup

To ensure that your backup volumes are accurate and error-free, the **sysadmsh** backup menu includes an Integrity option. The volumes are checked to see if they are readable and the contents are listed.

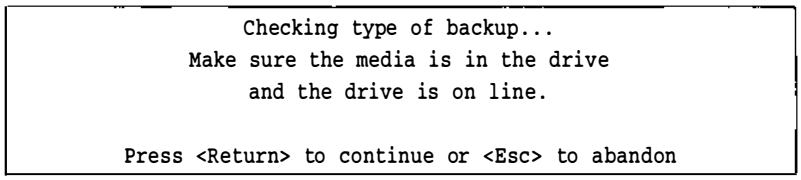
Invoke **sysadmsh** and select the following:

Backups ➪ Integrity

The following form is displayed:



Enter the Media type, or press ⟨F3⟩ to select it from a point-and-pick list. When selected, a window pops up to confirm the drive is ready:



Insert each volume of the backup in turn. This is a lengthy process which may take some time.

Getting a backup listing

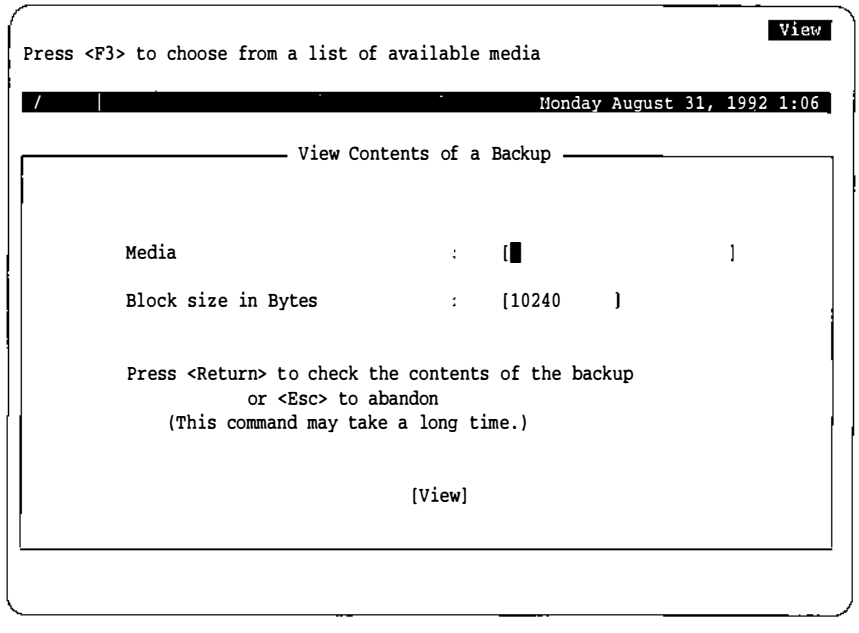
You can examine a list of the files you have backed up by generating a listing from the **sysadmsh** Backups menu.

To get the listing, follow these steps:

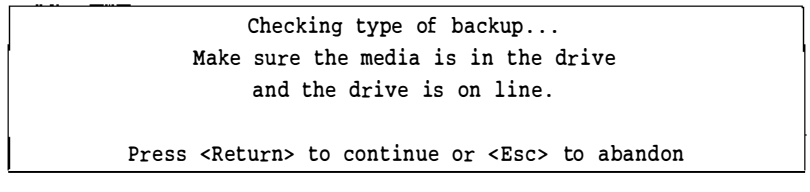
1. Select the following from **sysadmsh**:

   Backups ⇨ View

2. The following form is displayed:

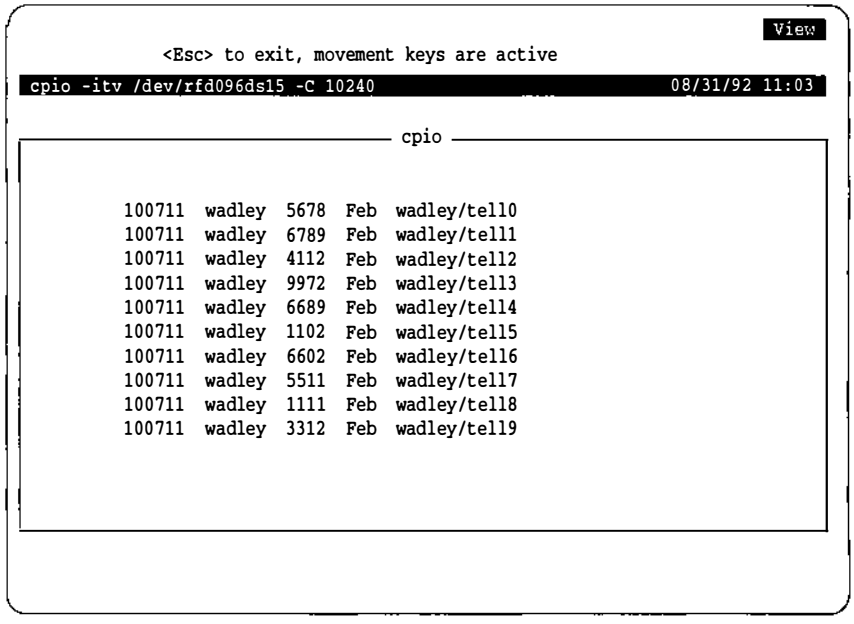


3. Press ⟨F3⟩ at the first field to get a listing of media devices. When you select a media device, a window pops up to confirm the drive is ready:



   The block size is selected automatically.

4. The program prompts you to insert each backup volume in turn.
5. When all volumes of the backup are read, a screen similar to the following is displayed:

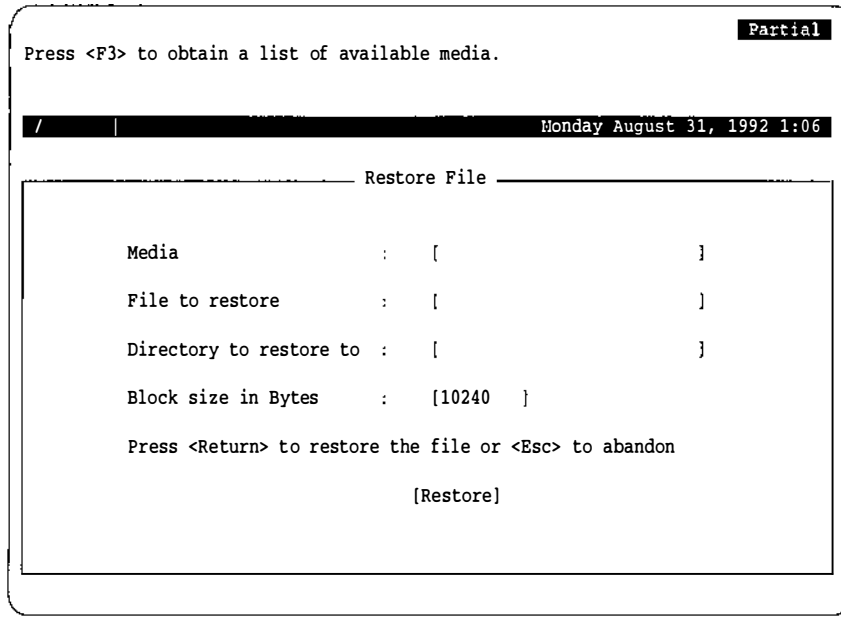


Restoring individual files or directories from backups

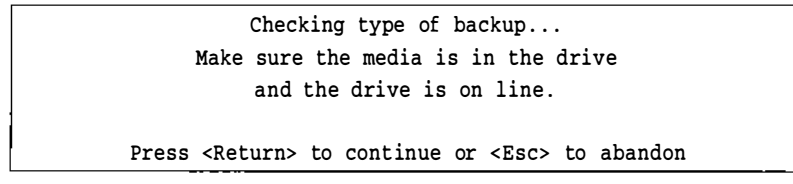
You can restore individual files or directories from your filesystem backup volumes by invoking **sysadmsh**. You need the complete set of backup volumes containing the latest version of the file or files you wish to restore. If you are restoring a file that was not changed recently, use the last level 0 backup.

To restore a file, follow these steps:

1. Invoke **sysadmsh** and select the following:
   Backups ⇨ Restore ⇨ Partial
2. You see the following:



3. Press ⟨F3⟩ first to select the Media type from a point-and-pick list. When selected, a window pops up to confirm the drive is ready:



4. Load volume 1 of the backup set into the drive, then press ⟨Return⟩. When this request is satisfied, you are returned to the "Restore File" menu. Enter the filename next, then press ⟨Return⟩ to move to the "Directory" field, entering the directory you wish to restore the file(s) to.

   NOTE Two important points:

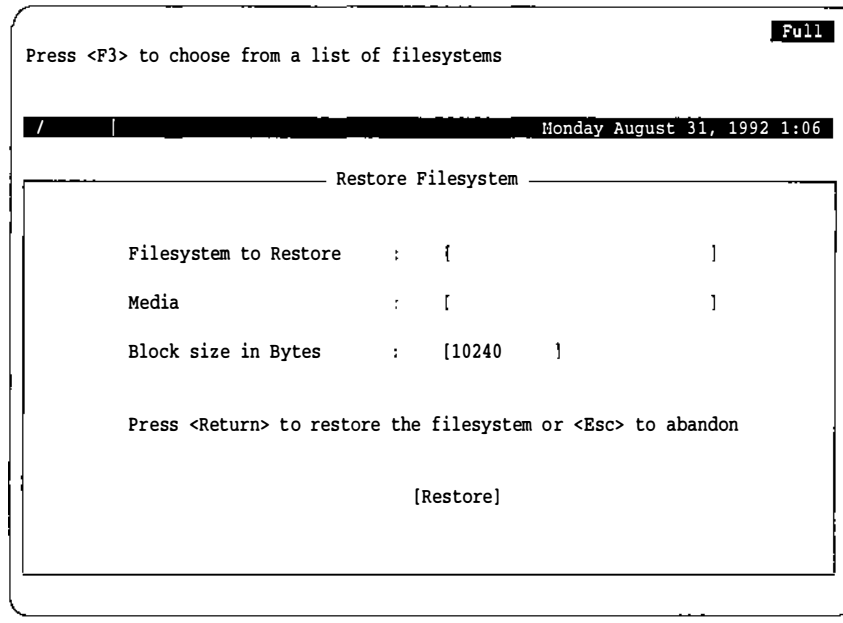
   - When specifying the pathname, the leading slash " / " must be removed. For example, if you are restoring the file */bin/foo*, you must specify it like this: **bin/foo**
   - If you respond with the pathname of the original location, the restored files overwrite any files by the same names in that location. It is important to be sure that the files on the backup volume are the desired versions of these files. If you are not absolutely sure that your backup contains the preferred version of the files, you should restore them to a temporary location, such as */tmp*, and compare them with your current files on disk using **diff**(C) or **cmp**(C).

5. The archive is searched for the files specified and the filename is displayed after it is restored to the specified locations on your hard disk. You are also prompted to switch volumes if necessary. If you know all the files you want were restored, you can exit the restore using the ⟨Del⟩ key. (Otherwise the program continues to search to the end of the backup.)

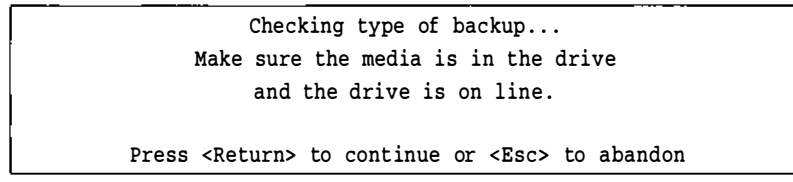
Restoring an entire filesystem

Follow these steps to restore your filesystem backup:

1. Insert the first volume, and make the following **sysadmsh** selection:
   Backups ⇨ Restore ⇨ Full
   The following form is displayed:



2. Enter the name of the filesystem, or press ⟨F3⟩ for a point-and-pick list. Do the same for the media device. The following window pops up to confirm the drive is ready:



3. You are asked to confirm that this is what you wish to do.

4. As each file is restored, the name is printed on the screen. If your backup has multiple volumes, you are prompted to insert each in turn:

```
Reached end of medium on input
Change to part n and press <Return> key. [q]
```

NOTE If **xrestore** is used to restore the filesystem, a slightly different prompt is displayed.

When the restoration process is complete, the number of blocks restored is displayed.

A detailed explanation of backup levels

The most straightforward and dependable way to ensure the safety of data is to back up everything on a filesystem at one time. However, filesystems can be large (as much as 200 Mbytes or more), and may take hours to back up. The concept of backup levels (or incremental backups) addresses this problem. The general idea of an incremental backup is to back up only those files that have changed since a previous backup. This can significantly reduce the size and duration of the backup. Consider the following scheme:

Monthly	complete backup
Weekly	everything newer than last week
Daily	everything newer than yesterday

This means that at the end of every month, the entire filesystem is backed up. Each week, the files that have changed since last week are backed up, and each day, any files that have changed since yesterday. If at some point a filesystem is damaged, you would simply restore the last full (monthly) backup, the last weekly backup, and any daily backups that happened just prior to the accident. Thus it is always possible to reconstruct a filesystem from a series of backups.

While this is a simple method to understand, the implementation using incremental backup levels is not.

Principles of incremental backup levels

To make the business of backing up files more efficient, the backup facility uses a progressive series of levels, each of which is based on the last occurrence of a lower-level backup.

Level	Files saved
0	all files on the filesystem
1	files changed since last level 0 backup
2	files changed since last level 1 backup
3	files changed since last level 0, 1, or 2 backup

The levels serve to subdivide a backup into manageable units. It is important to realize that each backup level creates backups based on the previous (next lowest) level backup. This means that the order of the backups is not significant, but the level number is.

For example, let's assume that the following backups were done for a week:

Day	Level	Files backed up
Mon	0	all files on filesystem
Tue	2	all files changed since Monday
Wed	1	all files changed since Monday
Thu	3	all files changed since Wednesday
Fri	2	all files changed since Wednesday

This example is illogical, but serves to demonstrate how the levels work. Remember that each of the backups saves the files changed since the next lower-level backup, and that level 0 is the lowest. Therefore, the level 2 on Friday backs up all files changed since the next lowest number, level 1, on Wednesday. The level 2 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.

How the default and alternate schedules work

The *schedule* file provided with your distribution is optimized for use on systems under moderate use (8 to 10 users with total disk storage of 200 to 400 Mbytes). The default schedule for the root filesystem is similar to that used for the */u* filesystem in the alternate schedule. This is done because a system with a single filesystem (*root*) has active user accounts and should be backed up each day. A system with a second filesystem (*/u*) for user accounts is backed up each day, while the less active root filesystem is backed up every other day. The alternate schedule is shown in Example 15-6.

Example 15-6 The default schedule

```
#                1 2 3 4 5    6 7 8 9 10    1 2 3 4 5    6 7 8 9 10
#    Filesystem  M T W T F    M T W T F     M T W T F    M T W T F
     /dev/rroot  0 x 3 x 3    2 x 3 x 3     1 x 3 x 3    2 x 3 x 3
     /dev/ru     3 0 3 3 3    3 2 3 3 3     3 1 3 3 3    3 2 3 3 3
```

The /u filesystem

Filesystem */dev/u* is a heavily used resource. Some level of backup is performed every day. This scheme is designed to minimize resources while maximizing safety; if one or more of the backups for that week is lost or goes bad, there is sufficient redundancy to minimize any loss of data.

According to the default schedule, a full (level 0) backup of */dev/ru* occurs at the beginning of the month. (Because a level 0 is done on the root filesystem on Monday, the level 0 for */u* is done on Tuesday.) On Wednesday, a level 3 backup saves just those files on */dev/ru* that have changed since the level 0 backup. By the end of the week far fewer floppies or tapes are used than the number needed for full backups each day. Time is substantially reduced as well. If it is necessary to restore the filesystem to the last recorded state, you would restore the last level 0 backup, followed by each of the most recent lower-level backups that were done since.

Note that each Tuesday, a lower-level backup (0, 1 or 2) occurs that saves everything since the beginning of the month and causes each of the level 3 backups that follow it to be based on that week. This way the level 3 backups do not become too large and redundant.

The root filesystem

The root filesystem contains the operating system and other system files. It changes less frequently, so it is not backed up every day unless user accounts are located there. Each Monday, a lower-level backup is done, and level 3 backups are done twice per week. Just as with the */u* filesystem, the level 3 backups are restricted to cover only those files that have changed during that week.

How backups restore a filesystem

For example, assume you have a hardware failure that ruins the information on the hard disk. Assume it happens on the last Thursday of the month, just before the backup was to be done that evening. You fix the hardware problem and reinstall your system, but how do you restore your backups? Restore the last occurrence of each backup level, in ascending order:

- level 0 (done on the first Tuesday of the month)
- level 1 (done on the third Tuesday)
- level 2 (done on the fourth Tuesday)
- level 3 (done on Wednesday evening)

You would not need to restore the level 2 that was done on the second Tuesday, because the level 1 that followed it covered the same files. The only information that is missing is what was changed during the day on Thursday, just before the crash. This is the primary reason for backups; recovery should be straightforward and with a minimum of loss.

Unattended backups

The methods for performing backups described so far require human operation. You may find it more convenient for your backups to be performed when the system is unattended, during early mornings for example. This can be achieved by making entries in root's *crontab* file to call the **cbackup** shell script at the desired times. If you are unfamiliar with the **cron** daemon and its use then examine the **cron**(C), and **crontab**(C) manual pages for an explanation of its function.

The cbackup shell script

This shell script is the last component that **fsphoto**(ADM) calls to perform a **cpio** backup. Using **cbackup** directly will bypass the schedule file, but still allow you to make incremental backups. **cbackup** takes four arguments:

level: the increment level of the backup you wish to make

size: the capacity of one volume of the media you are going to record the backup on

device: the name of the device to record the backup on. You should always specify a raw device.

filesystem: the name of the device of the filesystem to be backed up. **cbackup** looks up the mount point of the filesystem in */etc/default/filesys*. Since **cbackup** uses **cpio**, the filesystem must be mounted when it is backed up.

NOTE A single day's backups must fit on one volume for unattended backups to be successful.

Example crontab entries

If, for example, you wish to backup the entire root filesystem at 2:00 am every day on 150Mbytes cartridge tape, the following root *crontab* file could be used:

```
* 2 * * * /usr/lib/sysadmin/cbackup 0 150000 /dev/rct0 /dev/root
```

If you wish to perform a level 1 backup of the root filesystem every weekday morning and a full backup on Saturday morning then the following root *crontab* file could be used:

```
* 2 * * * 1-5 /usr/lib/sysadmin/cbackup 1 150000 /dev/rct0 /dev/root
* 2 * * * 6   /usr/lib/sysadmin/cbackup 0 150000 /dev/rct0 /dev/root
```

NOTE As mentioned before, **cbackup** ignores the schedule file. A hierarchy of incremental backups can be constructed by stating explicitly in the *crontab* file when a particular level should be performed.

If you wish to back up more than one filesystem to the same unattended device, then the backups have to run sequentially. To ensure this happens it is best to place the calls to **cbackup** in a shell script and make a **crontab** entry to call it. For example, if you wish to perform a level 1 backup of the */u* and *root* filesystems to a cartridge tape every weekday night then the following could be used:

crontab entry:

```
* 2 * * * 1-5 /usr/lib/sysadmin/bscript
```

/usr/lib/sysadmin/bscript:

```
/usr/lib/sysadmin/cbackup 1 150000 /dev/nrct0 /dev/root
/usr/lib/sysadmin/cbackup 1 150000 /dev/nrct0 /dev/u
tape rewind /dev/xct0
```

Notice the use of the no rewind tape device (*/dev/nrct0* here). It is required so that when the backup of the *root* filesystem has finished the tape is not automatically rewound to the beginning. For convenience you may wish to rewind the tape after the backup of */u* has finished.

Performance and Troubleshooting

Chapter 16

Troubleshooting overview

Troubleshooting is the general term for studying and resolving system problems. Most commonly, troubleshooting concerns correcting errors that led to a system crash or processes failing. It can also involve resolving performance problems (some of which may eventually cause the system to crash or processes to fail). The following chapter gives an overview of how to study and resolve performance problems; this chapter discusses methods for studying other system problems.

Chapter 20 (page 551) contains miscellaneous notes about problems that may arise, their causes, and how to correct them. Additional troubleshooting information is located in the *Installation Guide* which contains information about solving problems that arise when installing the UNIX software.

Avoiding system problems

Some system problems are unavoidable, but there are a number of actions you can take that will allow you to catch errors before they cause serious damage and to be able to recover from system problems quickly. These include:

- Always allow **fsck**(ADM) to run on all "dirty" file systems before they are mounted, especially after a system crash. **fsck** repairs a number of small filesystem errors before they cause serious damage.
- Monitor system performance regularly. This enables you to reallocate resources before they are exhausted and cause processes to fail or the system to crash.

- Monitor system error logs regularly so you can resolve problems before they cause serious system or process failure. The */usr/adm/messages* file contains all kernel and hardware messages written to the console since the file was purged. You can also use the XENIX **dmesg**(ADM) command to view all console error messages that appeared since the system was last booted. The **strace**(ADM) command displays all STREAMS error messages that have been posted.
- Keep a system log, and note all information that could be of interest in troubleshooting the system. Record all system errors that occur, all corrective and preventative actions taken, every time you change the hardware or software configuration, and so forth. For each event, log what happened, when it happened, who was involved, and any information you have about why it happened. Study this log periodically to spot trends that might indicate a developing problem.
- Always save a memory dump after a system crash. When you reboot the system, you may see the following message:

  ```
  There may be a system dump memory image in the swap device.
  Do you want to save a copy of it? (y/n)
  ```

  You can save this dump and study it with the **crash**(ADM) command. See "crash(ADM) — view memory" (page 505). We recommend that you archive dumps from previous crashes; this can provide a valuable history of developing problems. Old memory dumps can be archived to tape if necessary if you are short of disk space, but they must be written to disk before you can run **crash** on them.

  The system writes the memory dump to *dumpdev*, which is normally the same as *swapdev*. For the memory dump to be saved correctly, *dumpdev* must contain at least as much swap space as you have physical memory.
- Study the C2 security features carefully before setting access permissions.

Preparing to recover from problems

In addition to taking steps to avoid problems, you can take steps that will enable you to recover from system problems more quickly and smoothly:

- Back up the system regularly. Consider running full system backups every week or two. Because incremental backups are usually faster than full backups, it is tempting to run full backups only rarely. However, if you need to restore files, you may need to restore the most recent full backup plus all incrementals run since then, which could be a long, tedious process.

- Make the Emergency Boot Floppy Set as discussed in the *Installation Guide*. If the *root* filesystem or a critical booting file are corrupted by a system crash, the Emergency Boot Floppy Set can be used to recover the system. Otherwise, you may need to reinstall the software before you will be able to boot the system.
- Set up the system so **fsck**(ADM) fixes any filesystem problems whenever the system is rebooted. See "Repairing filesystems with fsck" (page 117) for more information.
- Maintain an "override" terminal as discussed in "Using the override terminal" (page 595). An override terminal can be used to correct errors you made in setting C2 access permissions that prevent access to other terminals. By default, the console multiscreen (*/dev/tty01*) is the override terminal.

Kernel error messages

Kernel error messages report driver errors and errors in other parts of the kernel such as the process scheduling subsystem and the file subsystem. These error messages appear on the console, and are usually recorded in the */usr/adm/messages* file. Monitoring these messages regularly is an important step in preventing serious system problems; studying these messages after a system problem develops is an important part of troubleshooting the system.

Kernel error messages usually have the format:

class: [*driver:*] [*routine*] *message*

class is usually one of the following:

CONFIG Identifies a problem that can be corrected by modifying the system configuration. Often, such messages indicate that a kernel resource under control of a tunable parameter has been exceeded. The *message* tells what the resource is and identifies the tunable parameter that controls its size. See "Kernel tables are too small" (page 493) for more information.

NOTICE An unexpected situation has occurred. Typically, **NOTICE** errors are less serious than **WARNING** errors in terms of system stability, although they may indicate an error that has serious ramifications for executing applications.

WARNING An error condition has occurred.

PANIC The kernel is corrupted and the system shuts down to prevent additional problems.

driver names the device having problems and is usually the same as the internal name defined in column 1 of *mdevice*(F). The actual peripheral is usually identified by a pair of numbers of the form ***major/minor***. This identifies the device number of the peripheral where the error occurred. The major and minor numbers also appear in the **ls -l** listing on a */dev* file, in the column just before the date where they byte count appears on regular files. For example, the following output shows that 2 is the major number and 52 is the minor number of */dev/fd0*:

```
brw-rw-rw-   7 bin      bin        2, 52 Jun 11 1990 /dev/fd0
```

The ***routine*** element indicates the subsystem that detected the condition; these portions of the error message are included mostly to help support staff in tracking difficult system problems.

The ***message*** explains what the error condition is. In most cases, this should be enough to identify the problem. The following examples illustrate kernel error messages:

`WARNING: floppy: Read error on dev 2/52, block=736 cmd=0x03 status=0x01`

This tells you that there was a read error on the floppy disk. Its severity depends on the importance of the data being read at the time. Usually this indicates that the floppy has a bad spot, although sometimes it indicates filesystem or disk corruption on the device that holds the data being read. When you get a read or write error on a device, you should verify that you have good data on the device; it may be necessary to repeat the copy operation.

`CONFIG: ***routine*** - Out of clists (NCLIST = 32 exceeded; 2 failures so far)`

This message tells you that you have exceeded a system resource (**clist**, in this case) that is controlled by a tunable parameter (**NCLIST**). It also tells you the current value of the tunable, and in this case, the number of overflow conditions that have occurred. You can use the index of this guide to locate information about the specific tunable parameter.

`NOTICE: ***routine*** - No space on ***fstype*** dev ***xnamex*** (***major/minor***)`

This message tells you that a filesystem has run out of space. ***fstype*** is the filesystem type (for example, EAFS, S51K, XENIX, or DOS). The ***major/minor*** information identifies the disk partition on which the filesystem is located, so by extension indicates the filesystem that has run out of space. This error message is discussed more in "Out of space on filesystem" (page 565).

PANIC: s5iread - Allocated *fstype* inode *num* on dev *xnamex* (*major/minor*) in free ilist
PANIC: s5iupdat - *filtyp* inode *num* address *num* > 2^24 on *fstype* dev *xname* (*major/minor*)

Error messages such as these two **PANIC** messages are not as easy to understand. ***xnamex*** is usually the same as the ***driver*** field discussed above. These two messages come from the file subsystem, which uses inodes to access a file. Normal conditions usually do not cause such problems, so messages such as these may indicate a bad memory board or some other hardware problem.

Recovering from a system crash

A system crash is any time the system goes down without closing files and doing other cleanup operations. Two types of system crashes occur:

panic — The system "panics" when it encounters a hardware problem or kernel inconsistency that is so severe that the system cannot continue functioning.

powerfail — If the AC power to the system fails, even briefly, the system crashes.

When your system goes down because of a panic or power failure, the shutdown is considered "abnormal." A third type of abnormal shutdown occurs when the operator must crash the system, usually because a program goes into a loop or the system hangs. When the system is hung because of an application program error, you should crash it as soon as possible; the longer the loop goes on, the greater the possibility of file damage. Recovering from an operator-induced crash is similar to recovering from a system panic.

When the system is shut down normally, the **shutdown**(ADM) program stops all daemons, kills the active processes, unmounts any mounted filesystems, and runs the **sync** command, and tells **init** to bring the system down to the appropriate state (either single-user or "safe to power off"), or to reboot.

If the system goes down before this shutdown procedure completes, the following may occur:

- Filesystems may be inconsistent or corrupted, resulting in lost data.
- Ongoing work by users and other data may be lost because the buffer cache was not flushed to disk.

UNIX System V synchronizes the disk with memory (**sync**'s) frequently, so the amount of data lost due to system shutdown without the **sync** command should be minimal. However, because the filesystems were not unmounted properly before the system shut down, filesystem corruption may be extensive. If the *root* filesystem is corrupted, the system may not function properly.

Recovering from a system panic

The system "panics" when it encounters a hardware problem or kernel inconsistency that is so severe the system cannot continue to function. When this happens, the system usually displays a message on the console, and all system activity stops.

The system panic messages begin with the word "PANIC," followed by a diagnostic message. Sometimes this diagnostic explains what caused the panic, but sometimes it only indicates that the kernel was corrupted and the actual cause of the corruption was a few instructions back. The system can also panic without displaying a PANIC message. When this happens, the system simply refuses to process any input from the system console and all other terminals.

To recover from a system panic:

1. Copy the full PANIC message (including CPU registers), if any, from the console screen to your system log book.
2. Power-cycle the machine and press ⟨Return⟩ at the "Boot:" prompt to reboot the system.
3. At the prompt to check the *root* filesystem, answer " y ".
4. Bring the system up in single-user mode and study the problem that caused the system to panic. If possible, fix the problem.
5. When the system prompts you, respond " yes " to save a copy of memory at the time of the PANIC. The system writes this copy to the *dumpdev* device, which is usually the same as the *swap* device, so the data will be overwritten as soon as any paging occurs on the system. If you respond " yes " to save a copy of memory, it is written to either floppy or tape. You can later copy this data to disk and use the **crash**(ADM) command to study the information. See "crash(ADM) — view memory" (page 505).
6. Run **fsck**(ADM) on those filesystems that were mounted when the system panicked. This happens automatically for all filesystems that are marked dirty when the system is brought up multiuser, but by running **fsck** manually, you can control the response to problems that are found; see "Repairing filesystems with fsck" (page 117) for information. Any filesystems that are not checked automatically can be checked by issuing the **fsck** command when the system is in single-user state. For example, use **fsck /dev/u** to check the */u* filesystem.
7. Verify the integrity of the security system. See "Verifying the security system after a crash" (page 487) for more information.

In rare cases, the operating system becomes corrupted and must be restored or reinstalled. Refer to "Restoring a corrupted root filesystem" (page 568) and "Boot troubleshooting" (page 551) for more information. In most cases, simply rebooting the system solves the problem. However, if your system consistently shuts down with the same PANIC message, you should fix the problem that is causing the system to panic.

Recovering after a power failure

When the power fails, turn the machine off. This minimizes potential damage to your system if the power fluctuates.

Once the power comes back on, restore your system using the following steps:

1. Turn on the computer and press ⟨Return⟩ at the "Boot:" prompt.
2. If the *root* filesystem is corrupted, the system prompts you whether to check the */dev/root* filesystem; enter " y " to invoke **fsck**(ADM). The system may display the following message:

   ```
   FREE INODE COUNT WRONG IN SUPERBLK
   FIX?
   ```

   This message is routine when the system is not shut down properly. If the system is not set up to automatically try to fix all *root* filesystem corruption (see "Repairing filesystems with fsck" (page 117) for more information), enter " y " and **fsck** fixes the problem.
3. If any non-root filesystems are corrupted when the system enters multi-user mode, the system prompts you whether to clean the filesystem; enter " y ". If the system is not set up to automatically fix filesystem corruption (see "Repairing filesystems with fsck" (page 117) for more information), you will be prompted each time **fsck** finds an error as to whether or not you want the corruption fixed.

Verifying the security system after a crash

A system crash may damage the Authentication database of files that control system security. Several programs are used to maintain the Authentication database, the system area of the filesystem, and the filesystem as a whole. The basic rule is to work from the most basic components of the filesystem outward. Otherwise, corrections made at the higher levels may be undone by programs fixing the lower levels.

After a system crash or abnormality, run the following programs in this order while the system is in single user (system maintenance) mode.

1. Run **fsck**(ADM) to check filesystem integrity (automatic when rebooting to multiuser state).
2. Run **tcbck**(ADM) to check for the absence and integrity of critical security files (automatic when rebooting to multiuser state). **tcbck** calls **authck**(ADM) to attempt to repair any file corruption it detects. Administrators can also run **authck** from the command line or with the Accounts ⇨ Check ⇨ Databases selection from **sysadmsh**. When the **authck** command is run from the command line, you are given the option of allowing **authck** to repair any inconsistencies it finds. If corruption is found when you check the security files with **sysadmsh**, you can execute **authck** from the command line or restore files if necessary.
3. Make the Accounts ⇨ Check ⇨ Password selection from **sysadmsh** to check the consistency of the */etc/passwd* and */etc/group* files. This selection does the same checking that is done when a new user is created. If any error messages are generated, they must be acted upon.
4. Check system file permissions with **integrity**(ADM). See "integrity(ADM)" (page 509) for more information.
5. Run **fixmog**(ADM) to fix file permissions inconsistences found by **integrity**. **fixmog** changes the owner, group and access permissions of files to match the information in the File Control database. Always use the **-i** (interactive) option to ensure that you can confirm any changes before they are made.

See Chapter 9 (page 201) for more information about the security system.

Chapter 17

Performance management

Performance management involves budgeting system resources so they are most appropriately allocated for the way you use your computer.

UNIX System V includes tools for studying how system resources are being used on your system. The most important of these tools is **sar**(ADM) which reports on usage statistics. See "sar(ADM)" (page 511) for general information about setting up and studying **sar**; details about how to use these statistics are in the appropriate sections of this book. Information about performance management tools is arranged alphabetically by tool in Chapter 18 (page 503).

General performance considerations

The way you use and administer your system can impact system performance. Some factors to consider are:

Run large, resource-intensive jobs during off-hours

Jobs such as backups, source-code builds, and database archive and reorganization operations can be run when most users are off the system.

- Use the **sar -u** report to determine hours when the system is normally quiet.
- Examine the files in *crontabs* and *atjobs* under the */usr/spool/cron* directory to see if tasks being run during prime time could be run at times when the system is idle.
- Tasks that transfer large amounts of data to or from a tape or networking device should be scheduled for off-hours whenever possible, because in many cases they degrade overall system performance when they run. Examples include backing up filesystems to tape and receiving the USENET news database.

- If the system seems "bogged down," run the **ps**(C) command to identify resource-intensive processes that are running. See if these processes could be run during quiet times, perhaps using **cron**(C) or **at**(C).

Reorganize filesystems periodically to reduce disk fragmentation

Fragmentation is a bigger problem when filesystems become more than 75% full. If possible, configure the disk partitions and arrange directories on disk so that filesystems maintain 15-25% of their blocks free at all times. Any time a filesystem has more than 90% of its blocks full, expect to suffer serious performance degradation. See "Reducing disk fragmentation" (page 567) for more information.

Repair hardware promptly

Faulty hardware can seriously degrade system performance. Monitoring the */usr/adm/messages* file for signs of faulty hardware can help you identify hardware problems that are developing. You can also monitor the **sar -y** report; when the "mdmin/s" field shows values greater than 0, it may indicate that modems, ports, or terminals need repair.

Keep user PATH efficient

The **PATH** environment variable lists the pathnames of all directories to be searched each time a command is executed. Before displaying "not found," the system must search every directory in **PATH**. These searches require both processor and disk time. All the shells that run on UNIX System V support path caching which reduces the impact of inefficient **PATH** definitions, but individual users may still improve response time by keeping the **PATH** line as short as possible, with the most commonly-accessed directories (usually */bin* and */usr/bin*) at the beginning of the line. Avoid including large directories other than */bin* and */usr/bin* in the **PATH** line; if large directories must be included, put them at the end of the line.

The default **PATH** is defined in the */etc/default/login* file; users may then redefine the default **PATH** in their *.login*, *.cshrc*, or *.profile* files. (See **environ**(M) for more information on environment variables.)

Understanding your performance goals

The first step in performance management is to study the performance of your system and determine where potential or real performance bottlenecks are. These are the areas where you should concentrate your efforts.

Table 17-1 summarizes the symptoms and possible solutions for these performance goals.

Table 17-1 Performance goals summary

Goals	Symptoms	Solutions
Excessive paging	**sar -q**: %swpocc > 20 **sar -w**: swpin/s > 1.00 **swap -l**: < 50% of swap free during peaks	Reduce size of kernel (tune table sizes and number of buffers) Increase swap space (**swap -a**) Add memory Move users to another machine
Kernel tables too small	Console error messages **sar -v** shows overflows	Tune tables to acceptable sizes
Improve disk performance	**sar -u**: %wio > 10%	Use ACER filesystems Balance file systems across disks and controllers or host adapters. Keep file systems < 90% full Reorganize directories Keep directories small Tune I/O parameters Add more disks Use faster disks, controllers, host adapters
	sar -b: %rcache < 90 **sar -b**: %wcache < 65	Increase buffers (**NBUF**) Tune name cache parameters

The following sections discuss these goals in detail.

Excessive paging

User processes execute in the portion of memory that is not occupied by the kernel, plus the swap area(s) configured on your disk(s). If the amount of main memory available for user processes is inadequate, pages of processes are written out to the *swap* device on disk. This transfer of pages between memory and the disk is costly in both disk and CPU overhead. To see whether this is causing degraded performance on your system, look at the following:

- page faults (vflt/s) greater than 100 on the **sar -p** report
- free memory (freemem) less than 100 pages on the **sar -r** report
- pages read from the swap device (swpin/s) greater than 1 and idle time (%idle) greater than 0 on the **sar -wu** report
- CPU utilization on the **sar -u** report. As a general guideline, if the percent of time the CPU was occupied by system overhead (%sys) is consistently higher than 60, or if the percentage of time waiting for physical I/O (%wio) is greater than 0, you may be paging excessively.

Excessive paging happens when you run out of memory, so that pages of executing processes must be paged out to disk memory. To correct the problem, you must free up some memory. Some ways to do this are:

- Reduce the size of the system buffer cache. System buffers occupy a large portion of the kernel. The number of buffers configured in the system is displayed when you boot the system, or can be viewed in the */usr/adm/messages* file. The **sar -b** report shows how the buffer cache is being used on your system; see "Tuning disk I/O parameters" (page 494) for more information about adjusting the size of the buffer cache.
- Reduce the number of STREAMS buffers by tuning the **NBLK*n*** parameters, especially if you are not using networking services or other applications that use STREAMS heavily. See "STREAMS data" (page 542) for more information.
- Look for other kernel tables that may be unnecessarily large. If the peak number of entries shown as being used on the **sar -v** report are consistently much less than table size, you can decrease table sizes and regain some memory. As a general rule, none of these tables should be tuned more than 15% higher than the highest level of usage. The entries in the process table (controlled by the **NPROC** tunable parameter) are fairly large, so reducing this tunable may help considerably. If you reduce the value of **NPROC**, you can also reduce the size of **NREGION** that controls the size of the region table to a value slightly more than 3***NPROC**.
- Check that processes are not allocating unnecessarily large shared memory regions or leaving unused shared memory regions allocated when they exit abnormally. Use the **ipcs**(ADM) command to check the status of shared memory regions; use the **ipcrm**(ADM) command to removed an unused shared memory region from the system. You can use **crash**(ADM) to determine which processes are using shared memory.
- Add more memory to the configuration.
- Schedule large jobs to be run at off-hours or move users to another machine.
- Tuning the kernel to increase the value of **GPGSLO** and **GPGSHI** may help reduce the impact of excess paging if the other solutions cannot be implemented. **GPGSLO** is normally set to about 1/16 of pageable memory and **GPGSHI** is normally set to about 1/10 of pageable memory. Setting these tunables to higher values makes the paging daemon more active, which helps reduce the thrashing (excessive paging) that occurs when memory is extremely tight. See "Processes, memory management, and swapping" (page 535) for a description of these parameters.
- A few pseudo-drivers can be removed to save space, but because this is risky and the total savings are usually under 100 kilobytes, instructions have been omitted from this book.

Kernel tables are too small

The kernel contains a number of tables that contain control information for executing processes. If these tables are too large, the kernel is unnecessarily large and you may have inadequate user space for executing processes. If these tables are too small, processes may fail or, in some cases, the system may panic.

When a table overflows, the kernel prints out a message such as the following:

```
CONFIG: timeout - Timeout table overflow (NCALL = number exceeded)
```

This identifies the table that has overflowed, the tunable parameter that controls the size of that table (NCALL in this case), and the current value of that tunable parameter.

The Table 17-2 summarizes the error messages that are produced when tables overflow, the tunable parameter to adjust to change the size of the table, and any tools that may be available to view usage statistics for the table.

Table 17-2 Table overflow error messages

CONFIG: *text* (*tunable* = *number* exceeded)

Text	Tunable	Tools
aio_breakup - AIO buffer table overflow	NAIOBUF	
aio_breakup - AIO request table overflow	NAIOREQ	
aio_memlock - AIO process table overflow	NAIOPROC	
aio_setlockauth - AIO lock table overflow	NAIOLOCKTBL	
allocreg - Region table overflow	NREGION	
dk_name - Diskinfo table overflow	NDISK	
dosiread - DOS inode table overflow	DOSINODE	
event - Event channel full	EVDEVSPERQ	
event - Event table full	EVDEVS	
falloc - File table overflow	NFILE	sar -v
hsiread - High Sierra inode table overflow	HSNINODE	
hsiread - High Sierra extent table overflow	HSNEXTENT	
id - Out of STREAMS	NQUEUE	crash strstat
ifreeget - Inode table overflow	NINODE	sar -v
MPX timeout - Timeout table overflow	NCALL	
newproc - Process table overflow	NPROC	sar -v
s5iread - *filesystem_type* inode table overflow	NINODE	sar -v
sp: spclose - Cannot allocate STREAMS block	NBLK????	
string: Configuration buffer full	MAX_CFGSIZE	

(Continued on next page)

Table 17-2 Table overflow error messages
(Continued)

stropen1 - Out of streams	NSTREAM	crash strstat
stropen2 - Out of streams	NSTREAM	crash strstat
swapdel - Total swap area too small	MINASMEM	
timeout - Timeout table overflow	NCALL	
xsd_alloc - XENIX semaphore table overflow	XSEMMAX	
xtinit - Cannot allocate xt link buffers	NUMXT	
kernel_function - Out of clists	NCLIST	

Tuning disk I/O parameters

Many performance problems are caused by disk I/O bottlenecks. To correct these problems, you may need to rearrange the filesystems and swap areas on your disks, add more disks to your configuration, or map out bad areas of the disk so the system does not attempt to access them. See "Disk problems" (page 562) for information about analyzing and correcting disk I/O bottlenecks.

Disk performance may also be improved by tuning the parameters that control the system buffer cache. All disk I/O that uses the filesystem uses the buffer cache as an intermediate storage area. Most UNIX systems are I/O intensive, so assisting the I/O system with some tuning provides large benefits for users, especially when running I/O intensive applications. The system buffer cache also occupies a large portion of kernel memory, so if you need to reduce the size of the kernel or are running applications that are not I/O intensive, you may improve performance by reducing the size of the buffer cache.

If you have sufficient memory, you may be able to improve I/O performance by modifying the following kernel parameters. If you are trying to reduce the size of the kernel, tuning the values of these parameters down may be a good starting place.

NBUF (and **MAXBUF**)

NBUF controls the amount of memory the system dedicates to the system buffer cache that caches disk read and write operations. Each caching buffer occupies 1K of RAM plus a bit of memory for the buffer header.

The number of buffers configured is displayed when you boot the system with a line similar to the following:

```
kernel: i/o bufs = xxxk.
```

This information is also recorded in the */usr/adm/messages* file. After running the system with this value for a while, you can study the **sar -b** report to understand how the buffer cache is being used on your system. Usually you want the read cache hit rate ("%rcache" on the **sar -b** report) to be between 90% and 95%, and the write cache hit rate ("%wcache" on the **sar -b** report) to be between 65% and 85%. Theoretically, a cache hit rate of 100% is ideal, but in most cases sizing the buffer cache to achieve 100% hit rates uses so much memory that it causes performance problems in other areas. To determine the optimal buffer cache size for your system, adjust **NBUF** slightly and monitor the performance on your system. See "sar -b" (page 513) for more information.

Use the following guidelines when adjusting the size of the buffer cache:

- **NBUF** should usually have a value that is 10-25% of your system RAM. The tradeoff is that using more RAM for buffering leaves less RAM for user programs and may therefore increase paging.
- Adjust the size of the buffer cache in small increments, and monitor system performance carefully to see if additional adjustments are warranted.
- On most systems, the kernel should not exceed 16 Mbytes. Do not set **NBUF** so high that the kernel size exceeds this value unless you have a disk controller that supports DMA operations above 16 Mbytes. If such a controller is configured, the "%adapter fts = d" line will appear on the boot screen.

NHBUF The **NHBUF** parameter controls the number of hash buckets used to hash the buffer cache. Hash buckets are arrays of buffers that are used to speed up searches in the buffer cache. Check the hardware screen that is displayed at boot time for the size of the buffer cache and set **NHBUF** to a power of 2 that is approximately 1/4 that size.

S5CACHEENTS The **S5CACHEENTS** parameter controls the size of the **namei** cache that is used to speed the time searching through all directories in a path name when a file is first accessed. This caching speeds up the time required to locate a file. A good starting point for **S5CACHEENTS** is 3***NINODE**. If many users are accessing the same set of files, the **namei** cache can be smaller than if the users are accessing different files.

S5HASHQS **S5HASHQS** controls the size of the hash table that is used with the **namei** cache. This hash table works much like the buffer cache hash table discussed above. **S5HASHQS** should be set to a prime number that is roughly 1/4 the value of **S5CACHENTS**.

S5OFBIAS **S5OFBIAS** controls the length of time the system spends searching the **namei** cache for a free entry before it takes an entry that is in use (thus deleting the entry from the cache). The larger the value, the longer the search, but the more likely that an unused entry will be found rather than displacing a current entry. Typically, **S50BIAS** is set to 1/10 the value of **S5CACHEENTS**, and it must not be any larger than **S5CACHEENTS**.

For information about interpreting the **sar**(ADM) reports that report information to use when tuning the I/O subsystem, see "sar -b" (page 513), "sar -n" (page 517), and "sar -d" (page 515).

Optimizing serial I/O

This section covers some configuration changes that you can make to optimize serial throughput using the SCO serial driver on standard serial ports and non-intelligent cards. In particular, by changing your configuration, you can use the 16550 FIFO support in the SCO serial driver to obtain sustained data transfer rates greater than 9600 bps.

To support higher transfer rates, you should do the following.

NOTE The optimum configuration varies from system to system; you might need to experiment to determine the configuration that works best for your system.

1. Increase the value of the **TTHOG** kernel parameter.

   TTHOG is the maximum size of the raw queue of the tty driver. Increasing the value of this parameter allows more unprocessed characters to be retained in the tty buffer, which may prevent input characters from being lost if the system is extremely busy.

   The default value of **TTHOG** (256) is too small for high speed throughput. We recommend that you start by increasing **TTHOG** to 2048. Some applications might need a higher value, such as 4096.

   NOTE If you increase the value of **TTHOG**, you must also increase the value of the **NCLIST** kernel parameter (see step 2).

2. Increase the value of the **NCLIST** kernel parameter.

   Each serial port consumes 64-byte cblocks until it has used the number of bytes specified by **TTHOG**. If you increase the value of **TTHOG** to 2048, each serial port can use up to 32 cblocks. Therefore, you should increase the number of cblocks (by increasing the **NCLIST** kernel parameter) by 32*(number of serial ports). If you increase **TTHOG** to 4096, you should increase **NCLIST** to 64*(number of serial ports).

3. Change the default values in the **`sio_fifoctl`** kernel array.

   For serial ports using the 16550AFN UART chip, you can change the receive interrupt trigger level from 14 to 8 by changing the default values in the **`sio_fifoctl`** array. This causes the 16550 to interrupt sooner, so that a longer interrupt latency can be tolerated without character loss.

   Possible values for the interrupt trigger level are 14, 8, 4, and 1. A value of 8 is a good starting point; however, some applications might require a value of 4 or even 1. Lower interrupt trigger levels are appropriate when the kernel has a high maximum "interrupt latency". (The interrupt latency is the amount of time it takes the kernel to begin servicing the interrupt.)

Increasing the value of TTHOG and NCLIST

To increase the value of **TTHOG** and **NCLIST**:

1. Log in as *root* and bring the system down to system maintenance (single-user) mode. See "Entering system maintenance mode directly" (page 46).
2. Enter the following commands:

   cd /etc/conf/cf.d
   ./configure

   Δ **sysadmsh** users select: System ➪ Configure ➪ Kernel ➪ Parameters
3. Select "2. Character Buffers" from the menu.
4. Press ⟨Return⟩ until you see the **TTHOG** prompt.
5. At the **TTHOG** prompt, enter the new value for **TTHOG** (either 2048 or 4096, depending on your application).
6. Press ⟨Return⟩ until you return to the **configure** menu.
7. Select "2. Character Buffers".
8. Press ⟨Return⟩ until you see the **NCLIST** prompt.

9. At the **NCLIST** prompt, enter the new value for **NCLIST**.

   Use the following formulas to calculate the new value for **NCLIST**:

   TTHOG = 2048 new **NCLIST** value = old **NCLIST** value + 32*(number of serial ports)

   TTHOG = 4096 new **NCLIST** value = old **NCLIST** value + 64*(number of serial ports)

10. Press ⟨Return⟩ until you return to the **configure** menu.
11. Enter " q " to quit **configure**.
12. Enter " y " to update the system configuration files.

The **TTHOG** and **NCLIST** kernel parameters are now updated. For these changes to take effect, relink the kernel; see "Relinking the kernel" (page 501).

Changing the sio_fifoctl array

In SCO UNIX System V Release 3.2 Version 4, the **sio_fifoctl** array is initialized in the */etc/conf/pack.d/sio/space.c* file. You can edit *space.c* with the **vi**(C) editor.

To change the **sio_fifoctl** array in the *space.c* file:

1. Determine the elements of the **sio_fifoctl** array.

   The elements of the **sio_fifoctl** array correspond to the minor numbers of the non-modem control ports and the minor number minus 128 of modem control ports:

   non-modem control port

   To determine the minor number for a non-modem control port, display a long listing of that port in the */dev* directory. The minor number is the integer immediately before the date. Use this number for the **sio_fifoctl** array index.

   For example, to determine the minor number for the tty1a control port, enter:

   l /dev/tty1a

   You see:

   ```
   crw-rw-rw-   1 bin      bin        5,  0 Jan 22 1992 /dev/tty1a
   ```

   In this case, the minor number is 0; so you must modify the first array element (**sio_fifoctl[0]**).

modem control port

To determine the minor number for a modem control port, display a long listing of that port in the */dev* directory and subtract 128 from the minor number (the integer immediately before the date). Use this number for the **sio_fifoctl** array index.

For example, to determine the minor number for the tty2A modem control port, enter:

l /dev/tty2A

You see:

```
crw-rw-rw-   1 bin      bin        5,136 Jan 22 1992 /dev/tty2A
```

In this case, the minor number is 136; 136 minus 128 is 8, so you must modify the 9th array element (**sio_fifoctl[8]**).

NOTE The minor number for tty2A might be different on your system.

2. Enter the following commands:

cd /etc/conf/pack.d/sio
cp space.c space.c.old

3. Edit *space.c* to make the desired changes to the **sio_fifoctl** array:[1]

NOTE If you are using a port as a mouse port, that element of the **sio_fifoctl** array is zeroed. For example, if you are using tty1a as a mouse port, line 156 in your *space.c* file looks like this:

```
0,
```

```
146 /*
147  * 16550 FIFO control by minor dev: the high two bits are the RX fifo
148  * trigger level, and the low four bits are the size-1 to use for the
149  * TX fifo. Statically initialize the array here, so that buffering
150  * of individual channels can be disabled, e.g., for serial mouse.
151  *
152  * NB: The format of this array must not be changed, since it is edited
153  * by the 'mkdev mouse' script, which relies on having one entry per line.
154  */
155      unsigned char sio_fifoctl[CHANNELS] = {
156      FCR_Rx14 | 15,  /* <- This is the  1st array element, sio_fifoctl[0] */
157      FCR_Rx14 | 15,  /* <- This is the  2nd array element, sio_fifoctl[1] */
158      FCR_Rx14 | 15,  /* <- This is the  3rd array element, sio_fifoctl[2] */
159      FCR_Rx14 | 15,
160      FCR_Rx14 | 15,
161      FCR_Rx14 | 15,
162      FCR_Rx14 | 15,
163      FCR_Rx14 | 15,
164      FCR_Rx14 | 15,  /* <- This is the 9rd array element, sio_fifoctl[8]  */
165      FCR_Rx14 | 15,
166      FCR_Rx14 | 15,  /* FCR_Rx1, FCR_Rx4, FCR_Rx8, and FCR_Rx14 are       */
167      FCR_Rx14 | 15,  /* defined in <sys/siohw.h>.  They set the interrupt */
168      FCR_Rx14 | 15,  /* trigger level to 1, 4, 8, and 14 respectively.    */
169      FCR_Rx14 | 15,  /*                                                  */
170      FCR_Rx14 | 15,  /* 14 is the default trigger level                   */
171      FCR_Rx14 | 15,  /*                                                  */
172      FCR_Rx14 | 15,  /* The "15" means to use a size of 14 for the TX     */
173      FCR_Rx14 | 15,  /* fifo.                                            */
174      FCR_Rx14 | 15,
175      FCR_Rx14 | 15,
176      FCR_Rx14 | 15,
177      FCR_Rx14 | 15,
178      FCR_Rx14 | 15,
179      FCR_Rx14 | 15,  /* <- This is the 24th array element, sio_fifoctl[23] */
180 };
```

1. The comments to the left of the "FCR_Rx14 | 15" lines do not appear in the *space.c* file; you do not need to add them.

4. Modify the appropriate line in the *space.c* file. For example:

   sio_fifoctl[0] (the usual entry for tty1a/A)

   - To change the interrupt trigger level to 8, change line 156 to:

     ```
     FCR_Rx8  | 15,
     ```

   - To change the interrupt trigger level to 1, change line 156 to:

     ```
     FCR_Rx1  | 15,
     ```

   sio_fifoctl[8] (the entry for tty2a/A, as calculated in step 1)

   - To change the interrupt trigger level to 8, change line 164 to:

     ```
     FCR_Rx8 | 15,
     ```

   - To change the interrupt trigger level to 1, change line 164 to:

     ```
     FCR_Rx1 | 15,
     ```

   The changes made to *space.c* do not take effect until you relink the kernel; see "Relinking the kernel" (this page).

Relinking the kernel

To relink the kernel:

1. Log in as *root* and bring the system down to system maintenance (single-user) mode. See "Entering system maintenance mode directly" (page 46).
2. Enter the following commands:

   cd /etc/conf/cf.d
   ./link_unix

3. At the prompt to specify that the new kernel boot by default, enter " y ".
4. At the prompt to rebuild the kernel environment, enter " y ".
5. Enter:

   init 6

   This reboots the system with the new kernel.

Chapter 18

Tools reference

The operating system includes tools that measure performance, reconfigure the system for better performance, and analyze other system problems that arise. The system administrator can use these tools to locate problem areas. The tools described in this section are:

sar samples the state of the system and provides reports on various system-wide activities.

swap with the **-l** option, reports on the amount of available swap space. Use the **-a** option to configure additional swap devices.

crash displays various kernel structures. The discussion here concentrates on using **crash** to view how STREAMS resources are being used, to see the current value of many tunable parameters, and to study the state of the system at the time of a system PANIC. **crash** has many other uses that are not discussed here.

ps reports on processes currently executing.

timex reports on system resource usage during the execution of a command or program.

configure a menu-driven program used to modify the value of tunable parameters that are included in UNIX.

fsck checks the integrity of a filesystem and attempts to fix any corruption it finds.

idtune used to modify some tunable parameters that cannot be modified with **configure**.

integrity checks for missing system files or incorrect permissions on system files and directories.

configure(ADM) — reallocate kernel resources

The **configure** utility is a menu-driven program that presents each tunable kernel parameter and prompts for modification. After modifying kernel parameters, you must relink the kernel by invoking **link_unix**(ADM), reboot, and test the new kernel. To change any kernel parameter, do the following:

1. Reboot and enter single-user (maintenance) mode.
2. Enter the following commands to start the **configure** program:

   cd /etc/conf/cf.d
   ./configure

   Δ **sysadmsh** users select: System ➪ Configure ➪ Kernel ➪ Parameters
3. The **configure** menu is displayed. The parameters are grouped by category. The meanings of the parameters are discussed in Tunable system parameter descriptions (page 527).

   Choose a category by entering the number preceding it. The resources in that category are displayed, one by one, each with its current value. Enter a new value for the resource, or to retain the current value, simply press ⟨Return⟩. After all the resources in the category are displayed, **configure** returns to the category menu prompt. Return to the Main Menu to choose another category or exit **configure** by entering "q".

   NOTE You must have the package associated with the parameter to make use of the resource associated with it. For example, the STREAMS package must be installed for the STREAMS parameters to have effect. See the *Installation Guide* for information on installing UNIX system packages.
4. After you finish changing parameters, you must link them into a new kernel. Enter the following command:

   ./link_unix

   Δ **sysadmsh** users select: System ➪ Configure ➪ Kernel ➪ Rebuild

   This assembles each of the kernel modules into a new kernel, which must now be installed. Linking can take a few minutes.
5. Boot the new kernel with the following command:

   /etc/shutdown

   Δ **sysadmsh** users select: System ➪ Terminate

   A "Boot:" prompt appears. When you press ⟨Return⟩ to reboot the system, the new kernel is loaded and run.

If problems exist with the new kernel, reboot */unix.old* by entering "unix.old" at the "Boot": prompt.

Using the configure command line

configure also has a command-line interface suitable for use by application developers. For instance, a database developer who finds that 70 files rather than 50 files need to be locked simultaneously may provide a shell script to perform the reconfiguration. To find the current value of any configurable resource using the command-line interface, enter:

./configure -y *RESOURCE*

Here *RESOURCE* is the name of the tunable parameter (in uppercase). To change the value of any resource from the command line, enter:

./configure *RESOURCE=value*

Overriding configure limit warnings

The **configure** utility notifies you if you exceed certain limits on values for kernel parameters. This prevents you from accidentally exceeding a value, which could severely degrade performance in some cases. The **configure** menu program verifies that you do mean to override the limits. You can also use the **-o** option as described on the **configure**(ADM) manual page to override a configuration limit. The override option only works if you are specifying a parameter on the command line; you cannot use it with the menu.

crash(ADM) — view memory

The **crash**(ADM) utility allows you to analyze the core image of the operating system. It is frequently used in postmortem analysis of a system PANIC, but can also be run on an active system. **crash** is not primarily an administrative tool; it is most useful for debugging the kernel after installing drivers and other kernel-level code. Nonetheless, **crash** provides some information that can help you monitor the performance of your system and troubleshoot system problems that arise. This documentation provides information about **crash** reports related to these topics; see the **crash**(ADM) manual page for full information about using **crash**.

Initializing crash

You must be logged in as *root* to run **crash**. The full command to begin the **crash** session is:

`/etc/crash [-d` ***dumpfile*** `-n` ***namelist***`] [-w` ***outputfile***`]`

The dumpfile is */dev/mem* for an active system (the default), or the name of the file that contains the memory dump you saved when rebooting the system after a system PANIC. ***namelist*** is the name of the bootable kernel image, usually */unix* unless you booted the system under a different name.

When **crash** is initialized, you see the following display:

```
dumpfile= /dev/mem, namelist= /unix, outfile= stdout
>
```

The " > " is the **crash** prompt; enter the **crash** function you want followed by any options you need and ⟨Return⟩. Note the following:

- The **-w** option redirects the output of the command to the file specified. For example, **-w var_stats** writes the output of **crash var** to the *var_stats* file. You can use **-w** on the **crash** command line to redirect all **crash** output to a file, or you can use **-w** on the function line to redirect the output of that particular function to a file.
- Any **crash** function can be piped to any shell function. For instance, the command **var ! cat >>** ***file*** takes the output of the **crash var** function and appends it to the end of ***file***; **var ! lp** redirects the output of this same command to the line printer. Note that the exclamation point (!) serves as the first pipe symbol from within **crash**; use " ! " thereafter.
- Use the **q** function to end the crash session.

The **crash stat** command is a good starting place when analyzing a postmortem dump. **stat** lists off system statistics (such as the system name and time of PANIC) that you can use to verify that you are looking at the right dump. The **crash panic** command will usually contain the last PANIC message that was displayed on the console.

crash strstat (statistics about STREAMS buffers)

Use **crash strstat** to view current usage statistics on the STREAMS buffers for your system. This information can be used to select correct values for the **NBLK*n*** tunable parameters and other tunable parameters discussed in "STREAMS data" (page 542). The following example shows a sample session using the **crash strstat** utility. The listings are associated with the following parameters:

streams	**NSTREAM**
queues	***NQUEUE***
data block size ***n***	***NBLKn***

The listing "message blocks" is a composite number based on the total block allocation; it does not correspond to a kernel parameter. The **NBLK*n*** data blocks are also known as STREAMS buffers.

Sample crash strstat output

```
# /etc/crash
dumpfile = /dev/mem, namelist = /unix, outfile = stdout
> strstat
> ITEM                  CONFIG    ALLOC    FREE        TOTAL     MAX    FAIL
streams                   168       84       84         1190      86       0
queues                    624      380      244         2621     388       0
message blocks           1455      155     1300      1332119     323       0
data block totals        1164      155     1009      1079923     322     198
data block size    4      256        0      256        28571      19       0
data block size   16      168        2      166        10540      52       0
data block size   64      296       29      267       831712      90       0
data block size  128      168       87       81       152605     139     185
data block size  256      104       29       75        18593      83      13
data block size  512       72        8       64        31949      10       0
data block size 1024       48        0       48         2286       8       0
data block size 2048       40        0       40         3601       6       0
data block size 4096       12        0       12           66       1       0

Count of scheduled queues:   0
> quit
```

Table 18-1 defines the column headings used by the **strstat** display.

Table 18-1 STREAMS display headings

Heading	Description
CONFIG	number of items currently configured
ALLOC	number of items currently allocated
FREE	number of items available for allocation. This number is the difference between CONFIG and ALLOC.
TOTAL	number of attempted allocations since system startup
MAX	highest number of items allocated at one time since system startup.
FAIL	number of failures due to insufficient free items since system startup

A non-zero value in the "FAIL" column indicates that a request was made for a buffer and that no buffers of that size were available. STREAMS failures cause problems with TCP/IP, degrade system performance and, in extreme cases, can cause your system to lock up. In this situation, use the **configure**(ADM) command to increase the number of buffers.

var (values of tunable parameters)

The **crash var** command lists the values of the tunable parameters that are listed in the **var** structure. These same values are listed when you run **configure**(ADM) or **sysdef**(ADM), but the **crash** listing is useful when you want to check a value when you are running **crash**.

fsck(ADM) — check and repair filesystems

The **fsck**(ADM) program should be run on all filesystems after any system crash or abnormal system termination. **fsck** checks filesystem integrity. Because filesystems are not unmounted gracefully when the system crashes, the structure of the filesystem itself as well as user, system, and application files that were in the process of being updated can be damaged. In most cases, **fsck** fixes any filesystem problems; in some cases, **fsck** may be unable to fix a file, but can recover some portion of the file in the *lost+found* directory of the filesystem. The filesystem being checked must be unmounted when you run **fsck**; the system must be in single-user mode when running **fsck** on the *root* filesystem. If the filesystem is corrupted to a point where **fsck** is unable to recover it, you may be able to fix the filesystem manually with **fsdb**(ADM). See "Repairing a filesystem when fsck stops at size check" (page 571) for information. Otherwise, you may need to recreate the filesystem and restore the data from a backup.

Filesystems should always be checked after a system crash before they are mounted. Minor filesystem corruption can become widespread quickly if you continue to read and write a corrupted filesystem. **fsck** can be run with the **-y** option so that it will automatically attempt to fix any filesystem corruption it finds. **fsck -y** runs when rebooting the system if the appropriate string is in the files that control the booting process are set up appropriately:

Filesystem	String	File
root	FSCKFIX = YES	/etc/default/boot
all others	fsckflags = y	/etc/default/filesys

Without these lines, **fsck** reports each error it finds and waits for you to respond before it attempts to fix it during reboot. If these lines are not present and you have **AUTOBOOT=YES** in */etc/default/boot* (default), the system will not automatically reboot if any filesystems suffered any damage at all from the crash.

idtune(ADM) — reallocate kernel resources

The **configure**(ADM) command modifies parameters shipped with the system. When a program or application adds new entries to the *mtune* file, these parameters are not available via the **configure** command. You should use the **idtune**(ADM) command instead.

To change the value of a resource, enter the following commands:

cd /etc/conf/cf.d
../bin/idtune ***RESOURCE=value***

Here ***RESOURCE*** is the name of the tunable parameter (in uppercase) and ***value*** is the new value for that parameter. After using the **idtune** command, you should relink the kernel and shut down the system as with the **configure** command.

WARNING Never edit */etc/conf/cf.d/mtune*.

integrity(ADM) — check system file permissions

The **integrity**(ADM) program compares the entries of the File Control database against the actual file permissions on the system. If **integrity** finds inconsistencies, use the **fixmog**(ADM) command to fix the file permissions.

NOTE If your system is configured with the Low or Traditional security defaults, permission problems reported by **integrity** will have no effect on system operation.

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 some "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*.

ps(C) — check process activity

The **ps**(C) command obtains information about active processes. This command gives a "snapshot" picture of what processes are executing, which is useful when you are trying to identify what processes are loading the system. Things will probably change by the time the output appears; however, the entries that you should be interested in are **TIME** (minutes and seconds of CPU time used by processes) and **STIME** (time when the process first started). The following example contains sample output from the **ps -af** command.

Sample output from ps -af

```
    UID   PID  PPID  C    STIME TTY  TIME COMMAND
   root 22247 21299  0 16:54:41  T1  0:01 mscreen -n 2
johnson 22246 22079  0 16:52:53  T0  0:04 vi file2
   root 22285 22247  0 16:56:04  T1  0:06 mscreen -n 2
   root 22284 22247  0 16:56:04  T1  0:00 mscreen -n 2
markham 22274 22271  0 16:55:09  p1  1:05 rlogin colossus
markham 22271 22243  0 16:55:07  p1  1:05 rlogin colossus
 forbin 22304 21003  0 17:02:51 003  0:00 /usr/bin/mail kuprin
 fisher 22298 18505  2 17:02:29 011  3:28 nethack
   root 22305  1327 10 17:02:57  T2  0:00 ps -afe
```

NOTE If you booted your system from a file other than */unix* (such as */unix.old*), you must specify the name of that file with the **-n** option to **ps**. For example, **ps -ef -n unix.old**.

See "Runaway processes, stopping" (page 594) for information about using the **ps** command to perform such tasks as locating "runaway" processes (one that uses progressively more system resources over a period of time while you are monitoring it). You can also use **ps** to find processes that take a very long time to execute; you can consider using **cron**(C) to execute such jobs during an idle time of the day. "Managing processes" in the *User's Guide* explains how to do this.

sar(ADM) — system activity reporter

sar(ADM) provides information that can help you understand how system resources are being used on your system. This information can help you solve and avoid serious performance problems on your system.

Table 18-2 summarizes the **sar** options:

Table 18-2 sar options

Option	Reports on:
-a	file access system routines
-b	buffer cache activity
-c	system calls
-d	disk activity
-m	IPC message and semaphore activity
-n	name cache
-p	paging activity
-q	average length of run and swap queues
-r	unused memory pages and disk blocks
-u	CPU utilization
-v	process, inode, file, and lock table status
-w	system swapping and switching activity
-y	TTY device activity
-A	summary of all reports

The individual **sar** outputs are discussed in alphabetical order later in this chapter.

How sar works

Internal activity is measured by a number of counters contained in the kernel. Each time an operation is performed, an associated counter is incremented. The **sar**(ADM) utility generates reports based on the raw data gathered from these counters. **sar** reports can be used to diagnose system problems. The two most critical areas to monitor are memory and CPU usage. The functions monitored by **sar** are discussed in the subsections that follow, including analysis of sample **sar** output. **sar** can either gather system activity data "live" or extract information collected in data files created by **sadc** (System Activity Data Collector).

By default, the following **crontab** entry is installed in *sys* file in the */usr/spool/cron/crontabs* directory:

```
0 * * * 0-6 /usr/lib/sa/sa1
20,40 8-17 * * 1-5 /usr/lib/sa/sa1
5 18 * * 1-5 /usr/lib/sa/sa2 -s 8:00 -e 18:01 -i 1200 -A
```

The **sa1** entries produce records every 20 minutes during working hours and hourly otherwise. The **sa2** entry writes a daily average report into the same file every hour during work hours. You can modify these entries as desired. The output files are in binary format (for compactness) and are stored in */usr/adm/sa*. The filenames have the format *sadd*, where ***dd*** is the day of the month. (See the **crontab**(C) and **sar**(ADM) manual pages for more information on modifying the default **crontab** file.)

Running sar

The basic syntax of the **sar**(ADM) command is as follows:

sar [*-option*] [*file*]

where:

option one of the options described in the following sections

file name of the file for the day whose statistics you want to view. For example, to view the **sar -v** report for the tenth day of the most recent month, the command is:

sar -v -f /usr/adm/sa/sa10

You can also run **sar** in "real time". To do this, specify the sampling interval in seconds. For example, to take a sample every 15 seconds, the command is:

sar -v 15

As shipped, the system allows any user to run **sar** in real time. However, the files in the */usr/adm/sa* directory are readable only by *root*. Change the permissions on the files in that directory to allow other users to access the **sar** data.

The examples in this section are not intended to represent benchmarks; they serve to illustrate how the output can be used. When tuning your system, it is recommended that you use a "benchmark" (a program used to evaluate the performance of a system) and have the system under normal load for your application.

sar -a — file access operations

The **sar -a** option reports the use of file access operations. The operating system routines reported are as follows:

iget/s — number of files located by i-node entry per second

namei/s — number of filesystem path searches per second. "namei" calls "iget", so "iget/s" is always larger than "namei/s".

dirbk/s — number of directory block reads issued per second

An example of **sar -a** output, with a 30-second sampling interval, follows:

```
unix unix 3.2 2 i386    07/18/92
12:41:40  iget/s namei/s dirbk/s
12:42:10       4       1       3
12:42:40       2       1       1
12:43:10       5       2       3
Average        4       1       3
```

The larger the values reported, the more time the kernel spends accessing user files. This indicates how heavily programs and applications are using the filesystem(s). In general, if the ratio of "iget/s" to "namei/s" is greater than 5 and "namei/s" is greater than 30, it may indicate that your filesystem organization is inefficient. Refer to "Checking free space on filesystems" (page 567) and "Reducing disk fragmentation" (page 567) for ways to solve the problem.

sar -b — buffer activity

The **-b** option reports the following buffer activity.

bread/s — average number of physical blocks read into the system buffers from the disk (or other block devices) per second

lread/s — average number of logical blocks read from system buffers per second

%rcache — fraction of logical reads found in buffer cache (100% minus the ratio of breads to lreads)

bwrit/s — average number of physical blocks written from the system buffers to disk (or other block devices) per second

lwrit/s — average number of logical blocks written to system buffers per second

%wcache fraction of logical writes found in buffer cache (100% minus the ratio of bwrit/s to lwrit/s)

pread/s average number of physical read requests per second

pwrit/s average number of physical write requests per second

The entries that you should be most interested in are the cache hit ratios %rcache and %wcache, which measure the effectiveness of system buffering. If %rcache falls below 90, or %wcache falls below 65, it may be possible to improve performance by increasing the number of buffers. If your application is I/O-intensive and you have a large memory configuration, you may want to tune the buffer cache so that %rcache is around 95 and %wcache is around 80. The number of buffers in the system buffer cache are controlled by the **NBUF** tunable parameter. See "Tuning disk I/O parameters" (page 494) for more information.

Technically, 100% hit ratio for the read cache should mean better performance than the 90% recommended here, but getting that last 10% of hits often requires so much memory that performance suffers because of the lack of user memory for executing processes.

An example of **sar -b** output follows:

```
unix unix 3.2 2 i386    07/18/92
16:32:57 bread/s lread/s %rcache bwrit/s lwrit/s %wcache pread/s pwrit/s
16:33:07       3      39      93       1      16      91       0       0
16:33:17       4      40      90       2      16      87       0       0
16:33:27       4      41      90       3       7      64       0       0
Average        4      40      91       2      13      84       0       0
```

This example shows that the buffers are not causing any bottlenecks, because all data is within acceptable limits.

sar -c — system call activity

The **-c** option reports system calls in the following categories:

scall/s all types of system calls per second, generally about 30 per second on a busy 4 to 6 user system

sread/s read system calls per second

swrit/s write system calls per second

fork/s fork system calls per second, about 0.5 per second on a 4 to 6 user system. This number increases if shell scripts are running.

exec/s exec system calls per second. If "exec/s" / "fork/s" is excessively large, look for inefficient **$PATH**'s.

rchar/s characters (bytes) transferred by read system calls per second

wchar/s characters (bytes) transferred by write system calls per second

Typically, reads plus writes account for about half of the total system calls, although this varies greatly with the activities that are being performed by the system.

An example of **sar -c** output follows:

```
unix unix 3.2 2 i386    07/18/92
18:33:04 scall/s sread/s swrit/s fork/s exec/s rchar/s wchar/s
18:33:35      38      16       6   0.03   0.03    6089    1638
18:34:05      39      16       4   0.07   0.07    6123    1602
18:34:35      38      17       5   0.17   0.17    6042    1704
Average       38      16       5   0.09   0.09    6085    1648
```

This report is of interest mostly for programmers who are testing application programs. If scall/s is greater than 300 over an extended period of time, it may indicate inefficient code for applications that are running.

Note that the XENIX system **vmstat**(C) command that is also supported on UNIX provides cumulative statistics about system calls executed (**vmstat -s**) and number of forks (**vmstat -f**).

sar -d — block device activity

The **sar -d** option reports the activity of block devices.

device name of the block device(s) that **sar** is monitoring

%busy percent of time the device was servicing a transfer request

avque average number of requests outstanding during the period of time (measured only when the queue is occupied)

r+w/s number of read and write transfers to the device per second

blks/s number of 512 byte blocks transferred to the device per second

avwait average time in milliseconds that transfer requests wait idle in the queue (measured only when the queue is occupied)

avserv average time in milliseconds for a transfer request to be completed by the device (for disks this includes seek, rotational latency, and data transfer times)

An example of **sar -d** is as follows:

```
unix unix 3.2 2 i386    02/18/92
13:46:28  device %busy avque r+w/s blks/s  avwait  avserv
13:46:58 wd-0        6   1.6     3      5    13.8    23.7
         f1-0       93   2.1     2      4   467.8   444.0
13:47:28 wd-0       13   1.3     4      8    10.8    32.3
         f1-0      100   3.1     2      5   857.4   404.1
13:47:58 wd-0       17    .7     2     41      .6    48.1
         f1-0      100   4.4     2      6  1451.9   406.5
Average  wd-0       12   1.2     3     18     8.4    34.7
         f1-0       98   3.2     2      5   925.7   418.2
```

Using this information, you can see how often each device is in use, how many requests (on average) are waiting for the disk, and how quickly the disk can find and transfer the data. The above example was taken while transferring data from hard disk (wd-0) to floppy disk (f1-0). The large difference between the numbers for the two devices illustrates the performance advantages of hard disks.

Note that queue lengths and wait times are measured while the queue had something on it. If %busy is small, large queues and service times probably represent the periodic **sync** efforts by the system to ensure that altered blocks are written to the disk in a timely fashion.

The optimum setup is to keep the %busy number high and the avque number low. To do this, balance filesystems and swap areas across all disks and controllers or host adapters. See "Disk load balancing" (page 563) for more information.

sar -m — interprocess communication

The **sar -m** option reports on interprocess communication activities. Message and semaphore calls are reported as follows:

msg/s number of message operations (sends and receives) per second

sema/s number of semaphore operations per second

An example of **sar -m** output follows:

```
unix unix 3.2 2 i386    02/18/92
15:16:58   msg/s  sema/s
15:17:32    12.3    29.2
15:18:02    11.8    26.8
15:18:32    18.4    23.1
Average     14.2    26.4
```

If you are not running application programs that use messages and System V semaphores, these figures will all be equal to 0.00. If you are using these interprocess communications facilities and either msg/s or sem/s is greater

than 100, the application is not using the system efficiently. See "Message queues" (page 539) and "Semaphores" (page 540) for information about tuning these parameters.

sar -n — name cache activity

The **sar -n** option reports the name cache statistics.

c_hits number of cache hits

cmisses number of cache misses

hit% hit to miss ratio as a percentage

An example of **sar -n** follows:

```
unix unix 3.2 2 i386    02/18/92

11:26:05  c_hits cmisses (hit %)
11:26:10       9       0 (100%)
```

The **namei** cache improves the time required to search a full pathname when first accessing a file. Generally, the higher the "hit %" field the better. If the "hit %" consistently falls below 90%, increase the value of **S5CACHEENTS** and associated parameters. See "Tuning disk I/O parameters" (page 494) and **S5CACHEENTS** (page 535) for more information.

sar -p — paging activity

The **-p** option reports paging activity. The following page rates are recorded.

vflt/s number of address translation page faults per second (valid page not present in memory)

pflt/s number of page faults from protection errors per second (illegal access to page) or "copy-on-writes", pflt/s generally consists entirely of "copy-on-writes"

pgfil/s number of vflt/s per second satisfied by a page-in from the file-system (each pgfil causes two lreads; see **sar -b**)

rclm/s number of valid pages per second that the system has reclaimed (added to list of free pages)

An example of **sar -p** output follows:

```
unix unix 3.2 2 i386    02/18/92
12:01:51 vflt/s pflt/s pgfil/s rclm/s
12:56:52  13.91   2.80   5.63   11.21
```

High values (over 100) for vlft/s can indicate that application programs are not efficient for a paging system (poor locality of reference), paging tunables need adjustment, or the memory configuration is inadequate for the system load.

The **vmstat**(C) command also reports statistics about paging activity. An example of the information provided by the **vmstat -s** command is:

```
  17968 free swap space
  12222 demand zero and demand fill pages
  25932 pages on swap
  44589 pages in cache
  28719 pages on file
  33791 protection fault
  84644 pages are freed
     23 success in swapping out a process
      0 fail in swapping in a process
     22 success in swapping in a process
     98 swapping out a region
     64 swapping in a region
 457461 cpu context switches
1870524 system calls
```

This output is representative of a system where excessive paging may be degrading performance.

sar -q — process throughput

The **sar -q** option reports the average queue length while the queue is occupied and percent of time occupied.

runq-sz run queue of processes ready to run

%runocc percentage of time the run queue is occupied

swpq-sz swap queue of processes to be swapped out

%swpocc percentage of time the swap queue is occupied

An example of **sar -q** follows:

```
unix unix 3.2 2 i386    02/18/92
11:00:56 runq-sz %runocc swpq-sz %swpocc
11:01:07     1.7      98     1.5      36
11:01:17     1.0      63     1.0      31
11:01:27     1.0      58     1.0      49
Average      1.3      74     1.2      39
```

If %runocc is greater than 90 and runq-sz is greater than 2, the CPU is heavily loaded and response time may be degraded. In this case, moving to a multiprocessing configuration might improve system response time. If %swpocc is greater than 20, more memory or fewer buffers would help reduce swapping/paging activity.

sar -r — page and swap usage

The **-r** option records the number of memory pages and swap file disk blocks that are currently unused. The following are recorded.

freemem average number of 4K pages of memory available to user processes over the intervals sampled by the command

freeswap number of 512-byte disk blocks available for process swapping

An example of **sar -r** output follows:

```
unix unix 3.2 2 i386    02/18/92
12:01:51 freemem freeswap
12:56:52     208     5848
```

sar -u — CPU utilization

The CPU utilization is listed by **sar -u** (or by the **sar** command with no options specified). At any given moment the processor is either busy or idle. When busy, the processor is in either user or system mode. When idle, the processor is waiting for input/output completion or has no work to do. The **-u** option of **sar** lists the percent of time that the processor is in system mode (%sys), user mode (%user), waiting for input/output completion (%wio), and idle time (%idle). An example of **sar -u** follows:

```
unix unix 3.2 2 i386    02/18/92
09:20:08        %usr      %sys    %wio    %idle
09:40:12           6         7       2       86
10:00:03           7         9       3       80
10:20:07          14        16      10       61
Average            9        11       5       76
```

In typical timesharing use, %sys and %usr are about the same value. In special applications, either of these can be larger than the other without anything being abnormal. A high %wio generally means a disk bottleneck. A high %idle, with degraded response time, may mean memory constraints; time spent waiting for memory is attributed to %idle.

The "idle" (percentage idle) column can also provide some insight into system performance. This figure is normally between 40 and 100 percent, even with a large number of active users. When this figure falls consistently below 30%, the chief competition for resources does not involve memory at all; the critical resource is raw processor power. (Run the **ps**(C) command to make certain that the excessive CPU usage is not due to a runaway process that is stealing every spare CPU cycle.)

If you are running a large number of users, it may help to switch to intelligent serial boards if you are using non-intelligent cards. Intelligent cards take some of the burden off the CPU rather than adding to the amount of work it has to do.

In addition, you should examine the files in */usr/spool/cron/crontabs* to see if jobs are queued up for peak periods that might better be run at times when the system is idle. Use the **ps** command to determine what processes are heavily loading the system. Encourage users to run large, non-interactive commands at off-peak hours. You may also want to run such commands with a low priority by using the **nice**(C) or **batch**(C) commands.

sar -v — system tables

The **-v** option reports the status of selected kernel tables. From this report you know when the size of these system tables need to be modified.

proc-sz — number of process table entries presently being used/allocated in the kernel (controlled by **NPROC**)

inod-sz — number of inode table entries presently being used/allocated in the kernel (controlled by **NINODE**)

file-sz — number of file table entries presently being used/allocated in the kernel (controlled by **NFILE**)

ov — number of times an overflow occurred (one column for each of the above three items)

lock-sz — number of record lock table entries presently being used/allocated in the kernel (controlled by **FLCKREC**)

The values are given as level/table size, meaning the current number of table entries in use and the size of the table. An example of **sar -v** follows:

```
unix unix 3.2 2 i386    02/18/92

17:36:05  proc-sz ov      inod-sz ov      file-sz ov    lock-sz
17:36:35  78/ 350  0    173/ 1050  0    153/ 1400  0     2/ 100
17:37:05  96/ 350  0    192/ 1050  0    169/ 1400  0     3/ 100
17:37:35  57/ 350  0    145/ 1050  0    146/ 1400  0     3/ 100
```

This example shows that all tables are large enough to have no overflows. Sizes could be reduced to save main memory space if these are the highest values ever recorded.

To see the actual number of these table entries being used currently, use the following command:

sar -v 1 1

The **pstat**(C) command provides similar information in a different format. To see the actual number of these table entries being used currently, you can also use this command:

```
pstat | grep '^[0-9]'
```

sar -w — swapping activity

The **-w** option reports swapping and context-switching activity. The following statistics are reported:

swpin/s number of transfers into memory per second

bswin/s number of 512-byte-block units (blocks) transferred for swap-ins (including initial loading of some programs) per second

swpot/s number of transfers from memory to the disk swap area per second. If greater than 1, memory may need to be increased or buffers decreased.

bswot/s number of blocks transferred for swap-outs per second

pswch/s process switches per second. This should be 30 to 50 on a busy 4 to 6 user system.

An example of **sar -w** output follows:

```
unix unix 3.2 2 i386    02/18/92
19:53:44 swpin/s bswin/s swpot/s bswot/s pswch/s
19:53:58     0.0     0.0     0.0     0.0      37
19:54:14     0.0     0.0     0.0     0.0      39
19:54:24     0.0     0.0     0.0     0.0      39
Average      0.0     0.0     0.0     0.0      38
```

This example shows that there is sufficient memory for the currently active users, because no swapping is occurring. Many system administrators aim for zero swapping, which can be accomplished by adding physical memory. Disk accesses are significantly slower than memory accesses, so minimizing swapping activity is important for good performance.

The **vmstat**(C) command also provides information about swapping and context-switching activity. Use **vmstat -s** to view this data.

sar -y — terminal I/O

The **-y** option monitors terminal device activities. If you have a lot of terminal I/O, you can use this report to determine if there are any bad lines. Activities recorded are defined as follows:

rawch/s input characters (raw queue) per second

canch/s input characters processed by canon (canonical queue) per second. Canonical processing recognizes interrupt characters (⟨Del⟩ by default), quit (⟨Ctrl⟩\), suspend (⟨Ctrl⟩**Z**), and editing characters such as ⟨Bksp⟩ and kill (⟨Ctrl⟩**U**). For example, if the user types:

ha⟨Bksp⟩**ello**

it increments the value of "rawch/s" by 7 and the value of "canch/s" by 5.

outch/s output characters (output queue) per second

rcvin/s receiver hardware interrupts per second

xmtin/s transmitter hardware interrupts per second

mdmin/s modem interrupts per second

Not all terminal drivers are written to produce these statistics. All SCO serial, console, and pseudo-tty drivers and most drivers produced by other vendors produce the rawch/s, canch/s, and outch/s statistics.

The number of modem interrupts per second (mdmin/s) should be close to 0. The ratio of xmtin/s to rawch/s and canch/s, and the ratio of rcvin/s to outch/s should be fairly constant. For non-intelligent I/O cards, these ratios should be close to 1:1. Intelligent I/O drivers move many characters per interrupt, and some drivers do not even use interrupts, so the ratios will be different. If the number of interrupts per transmitted character starts to increase dramatically, it usually indicates a bad line that is generating extraneous interrupts.

An example of **sar -y** output follows:

```
unix unix 3.2 2 i386    02/18/92
16:50:11 rawch/s canch/s outch/s rcvin/s xmtin/s mdmin/s
16:50:41     112      15     653     103     102       0
16:51:11     107       7     654     104     105       0
16:51:41      99       5     641      99     105       0
Average      106       9     649     102     104       0
```

swap(ADM) — check and add swap space

Swap space is disk storage that is part of virtual memory. When main memory is exhausted, pages of processes that are sleeping awaiting their turn to execute can be written out to a swap device, thus freeing up main memory for executing processes. The system comes configured with one swap device. Adding additional swap devices with the **swap**(ADM) command makes more memory available to user processes. Excessive paging degrades system performance, but augmenting the swap space is a way to make more memory available to executing processes without tuning the kernel.

The following command adds a second swap device, */dev/swap1*, to the system. The swap area starts 0 blocks into the swap device and the swap device is 16000 512-byte blocks in size.

swap -a /dev/swap1 0 16000

Use the **swap -l** command to see statistics about all the swap devices currently configured on the system.

Running the **swap -a** command adds a second swap device only until the system is rebooted. To ensure that the second swap device is available every time the system is rebooted, use a startup script in the */etc/rc2.d* directory. For example you could call it S09AddSwap.

timex(ADM) — examine system activity per command

The **timex** command times a command and reports the system activities that occurred on behalf of the command as it executed. Run without options, timex reports the amount of real (clock) time that expired while the command was executing and the amount of CPU time (user and system) that was devoted to the process. For example:

```
# timex command command_options
real     6:54.30
user       53.98
sys        14.86
```

Running **timex -s** is roughly equivalent to running **sar -A**, but it displays system statistics only from when you issued the command until the command finished executing. If no other programs are running, this information can help identify which resources a specific command uses during its execution. System consumption can be collected for each application program and used for tuning the heavily loaded resources. Other information is available if the

process accounting software is installed; see the **timex**(ADM) manual page for information. To enable process accounting, remove the comment characters from the command lines in the */etc/rc.d/6/kinit* file and reboot the system.

XENIX system tools

The **pstat**(C) and **vmstat**(C) commands that were originally developed for the XENIX operating system are also supported on UNIX. These commands provide much of the same information available through the **sar**(ADM) and **crash**(ADM) commands, but in a different format. See the manual pages for more information.

Chapter 19

Kernel parameter reference

Kernel parameters control the allocation of kernel resources. These resources are constantly being used, released, and recycled, and include:

buffers	cache of in-memory storage units that hold recently used data. (Buffers increase efficiency by keeping this data on hand and decrease reading from the disk.)
table entries	space in system tables that the kernel uses to keep track of current tasks, resources, and events
other parameters	other definable values that govern special resources (such as the number of multiscreens available or the quantity of semaphores)

The use of these resources is defined by certain limits known as *tunable kernel parameters*. These limits can be decreased or extended, sometimes at the expense of other resources. Each resource or limit is represented by a separate kernel parameter. Deciding how to best optimize the use of these resources is known as performance or kernel tuning. Each time these tunable parameters are changed, their new values must be relinked (recompiled) into the kernel so that the new limits take effect.

There are several reasons for reallocating system resources:

- You install additional hardware memory and thus have greater memory resources to allocate.
- Persistent error messages are displayed indicating that certain resources are used up, such as inodes or table entries.
- The system response time is consistently slow, indicating that other resources are too constrained for the system to operate efficiently (as when too little hardware memory is installed).

- Resource usage needs to be tailored to meet the needs of a particular application.

Before tuning your system, study the usage patterns for your system and decide which parameters need to be changed. "Understanding your performance goals" (page 490) provides suggestions to get you started. The best approach is to make small adjustments to the parameters, then study system statistics for a week or more and make additional adjustments. By making gradual adjustments, you reduce the chances of causing serious system problems.

If one of your performance goals is to reduce the size of the kernel (usually because the system is paging excessively), first concentrate on tunable parameters that control large structures. The following table summarizes the size ramifications for a select group of tunable parameters.

Parameter	Number of bytes
NBUF	1096 (1024 for the buffer + 72 for the header)
NPBUF	72
NHBUF	16
NCLIST	12
NINODE	76
NHINODE	8
NFILE	12
NMPBUF	72
NMPHEADBUF	72
S5CACHEENTS	72
NPROC	344 (also influences value of NREGION)
NREGION	68
NBLK*n*	up to 5120

Specialized applications often require the reallocation of key system resources for optimum performance. For example, users with large databases may find that they need to lock more files simultaneously than the current allocation of file locks permit. Users who have no need for specialized features such as message handling may find that they can get a slight performance boost by deallocating those features.

Most of the tunable parameters discussed in this chapter are defined in the */etc/conf/cf.d/mtune* file. To manually change the values of specific tunable parameters, use the **configure**(ADM) command as discussed in "configure(ADM) — reallocate kernel resources" (page 504) or the **idtune**(ADM) command as discussed in "idtune(ADM) — reallocate kernel resources" (page 509).

WARNING Never edit */etc/conf/cf.d/mtune*. Whenever the system modifies a kernel parameter, an alternate entry is created in */etc/conf/cf.d/stune*, which overrides *mtune* with values specific to your system. The *mtune* file always contains the values these parameters had when the system was first installed, and can be a valuable reference.

This chapter lists the tunable parameters by the **configure** categories, in the order they are listed on the screens rather than alphabetically. The categories are:

1. Disks and buffers
2. Character buffers
3. Files, inodes and filesystems
4. Processes, memory management and swapping
5. Clock
6. MultiScreens
7. Message queues
8. Semaphores
9. Shared data
10. System name
11. STREAMS data
12. Event queues and devices
13. Hardware dependent parameters
14. Security
15. Asynchronous I/O parameters

The following categories of tunable parameters that are listed at the end of this chapter cannot be changed using **configure** (use **idtune**(ADM) instead):

16. Boot load extension parameters
17. Multiprocessing parameters

Disks and buffers

NDISK number of disk drives attached to the system. This value is increased by **mkdev hd** if necessary when additional disks are installed. When the Diskinfo table overflows, the following message is displayed on the console:

```
CONFIG: dk_name - Diskinfo table overflow (NDISK = number exceeded)
```

NBUF number of 1K system buffers allocated for the system buffer cache at boot time. The system buffer cache is a data cache or memory array used as a temporary storage area between the disk and user address space when reading to or writing from mounted filesystems.

If **NBUF** is set to 0, the system calculates the size of the buffer cache to a size that does not exceed the value of **MAXBUF**. The size of the buffer cache is displayed as "kernel i/o/bufs" at boot time; see the */usr/adm/messages* file. Cache hit rate increases with the number of buffers. Cache hits reduce the number of disk accesses and thus may improve overall performance. Study the **sar -b** report for statistics about the cache hit rate on your system. See "sar -b" (page 513) for more information.

The system buffer cache typically contains between 100 and 600 buffers, but may contain 8000 or more buffers on a large server system. Each buffer is fairly large, so having an unnecessarily large buffer cache can degrade system performance because too little space is available for executing processes.

For optimal performance, adjust number of hash queues (**NHBUF**) when you adjust the value of (**NBUF**).

NPBUF specifies how many physical I/O buffers to allocate. One I/O buffer is needed for each physical read or write that is active.

NHBUF specifies how many hash queues to allocate for 1K buffers. These are used to search for a buffer given a device number and block number rather than a linear search through the entire list of buffers. This value must be a power of 2. The **NHBUF** value should be chosen so that the value **NBUF** divided by **NHBUF** is approximately equal to 4.

CTBUFSIZE size of the tape buffer in kilobytes. This static buffer is allocated at **init** time and used by some tape devices such as the Q1C02 (*ct*) and SCSI (*stp*) tapes. This should have a value of 32 to 256. The following are reasonable values for associated circumstances:

- 32K bare minimum: insufficient to stream
- 64K minimum to allow streaming (good for systems with little memory) or little tape use (performance is not critical)
- 96K a first reduction, if default uses too much memory
- 128K default: good tradeoff performance
- 192K a first increase, if default provides poor performance
- 256K maximum

MAXBUF maximum possible number of buffers in the buffer cache. This is the number of buffer description headers in the kernel. Fewer than this number of buffers may actually be autoconfigured by the kernel at boot time, depending on how much core is present. If **NBUF** is non-zero, then exactly **NBUF** buffers are configured, and there is no reason for **MAXBUF** to be larger than **NBUF**. If **NBUF** is 0, the kernel configures at most **MAXBUF** buffers automatically.

DMAABLEBUF number of transfer buffers for DMA requests over 16 megabytes. This must be a value of 4 to 128.

PLOWBUFS amount of buffer cache that is contained in the first 16 megabytes of RAM. It is expressed as a percentage and should be as high as possible, if you do not have a 32-bit controller (indicated by "fts=d" on the boot screen), to limit the number of copying requests made from buffers above 16 megabytes (see **NCOPYBUF**). The default value is 100%. This parameter needs to be changed only if you have more than 16 megabytes of RAM on your system.

NCOPYBUF number of buffers available for copying requests made from buffers above 16 megabytes to buffers below 16 megabytes, for non 32-bit controllers.

NAUTOUP specifies the buffer age in seconds for automatic filesystem updates. A system buffer is written to the hard disk when it has been memory-resident for the interval specified by the **NAUTOUP** parameter without a write operation, and the **bdflush** daemon process runs. So, if **NAUTOUP** is set to 10, a system buffer is written to the hard disk when **bdflush** runs after the

buffer has been memory resident for 10 seconds. Specifying a smaller limit increases system reliability by writing the buffers to disk more frequently and decreases system performance. Specifying a larger limit increases system performance at the expense of reliability.

BDFLUSHR specifies the rate in seconds for the **bdflush** daemon process to run, checking the need to write the filesystem buffers to the disk. The range is 1 to 300. The value of this parameter must be chosen in conjunction with the value of **NAUTOUP**. For example, it is nonsensical to set **NAUTOUP** to 10 and **BDFLUSHR** to 100; some buffers would be marked dirty 10 seconds after they were written, but would not be written to disk for another 90 seconds. Choose the values for these two parameters considering how long a dirty buffer may have to wait to be written to disk and how much disk-writing activity will occur each time **bdflush** becomes active. For example, if both **NAUTOUP** and **BDFLUSHR** are set to 40, buffers are 40-80 seconds old when written to disk and the system will sustain a large amount of disk-writing activity ever 40 seconds. If **NAUTOUP** is set to 10 and **BDFLUSHR** is set to 40, buffers are 10-50 seconds old when written to disk and the system sustains a large amount of disk-writing activity every 40 seconds. Setting **NAUTOUP** to 40 and **BDFLUSHR** to 10 means that buffers are 40-50 seconds old when written, but the system sustains a smaller amount of disk writing activity every 10 seconds. With this setting, however, the system may devote more overhead time to searching the block lists.

PUTBUFSZ specifies the size of a circular buffer, **putbuf**, that contains a copy of the last **PUTBUFSZ** characters written to the console by the operating system. The contents of **putbuf** can be viewed using **crash**(ADM).

PIOMAP determines the size of the map entry array used by the kernel programmed I/O (PIO) breakup routine. This routine allows device drivers to do programmed I/O of large data blocks at interrupt level by breaking the data blocks into smaller data units. Users should not modify this parameter.

PIOMAXSZ maximum number of pages to use at one time for programmed I/O. Users should not modify this parameter.

Character buffers

The following parameters control various data structure sizes and other limits in character device drivers provided with the operating system.

NCLIST specifies the number of character list buffers to allocate. Each buffer contains up to 64 bytes. The buffers are dynamically linked to form input and output queues for the terminal lines and other slow-speed devices. The average number of buffers needed per terminal is in the range of 5 to 10. Each entry (buffer space plus header) contains 72 bytes. When full, input and output characters dealing with terminals are lost, although echoing continues, and the following message is displayed on the console:

```
CONFIG: Out of clists (NCLIST = number exceeded)
```

NEMAP maximum number of I/O translation mappings. The value of this parameter should not be changed.

NUMXT number of layers a subdevice can configure to support bit-mapped display devices such as the BLIT or the AT&T 5620 and 730 terminals. When this number is exceeded, the following message is displayed on the console:

```
CONFIG: xtinit - Cannot allocate xt link buffers (NUMXT = number exceeded)
```

NUMSXT number of shell layers (**shl**(C)) a subdevice can configure.

NKDVTTY number of virtual terminals (ttys) supported by the console keyboard driver. Users should not modify this parameter.

NCPYRIGHT defines the size of a kernel data structure used to print vendor copyright messages when the system is booted. Users should not modify this parameter.

MAX_CFGSIZE
maximum size of configuration information saved by the **strings**(M) driver. This is the maximum size of information saved into the */dev/string/cfg* file described on the **string**(HW) manual page. If this limit is exceeded, the following message is displayed on the console:

```
CONFIG: string: Configuration buffer full (MAX_CFGSIZE = number exceeded)
```

PRFMAX maximum number of text symbols that the kernel profiler (*/dev/prf*, can properly process. See **profiler**(ADM) for information about the kernel profiler.

KDBSYMSIZE
size of the kernel debugger symbol table. Must have a value of 50,000 to 500,000.

TTHOG the size of the raw queue of the tty driver. The default value is 256. Increasing the value of this parameter allows more unprocessed characters to be retained in the tty buffer, which may prevent input characters from being lost if the system is extremely busy.

Files, inodes, and filesystems

NINODE specifies how many inode table entries to allocate. Each table entry represents an in-core inode that is an active file such as a current directory, an open file, or a mount point. Pipes, clone drivers, and XENIX semaphores and shared data also use inodes although they are not associated with a disk file. The file control structure is modified when changing this variable. The number of entries used depends on the number of opened files. The entries are normally in the range of 100 to 400, but could be much larger for large server systems. **NINODE** must be greater than or equal to **NFILE**. Each open file requires an inode entry in the inode table. If the inode table is too small, one of the following messages is displayed on the console:

```
CONFIG: ifreeget - Inode table overflow (NINODE = number exceeded)
CONFIG: s5iread - inode table overflow (NINODE = number exceeded)
```

When the inode table overflows, the specific request is refused. Although not fatal to the system, inode table overflow may damage the operation of various spoolers, daemons, the mailer, and other important utilities. Abnormal results and missing data files are a common result.

If the system consistently displays this error message, use **sar -v** command to evaluate whether your system needs tuning. The "inod-sz" column shows the number of inode table entries being used and the number of entries available in the table. The value for **NINODE** should be at least 10% greater than the number of active inodes at peak use time.

NHINODE specifies the size of the inode hash table; value must be a power of 2.

NFILE specifies how many open file table entries to allocate. Each entry represents an open file. The entry is normally in the range of 100 to 600, but could be much larger for large server systems. (**NFILE** must be less than or equal to **NINODE**.) The **NFILE** control structure operates in the same manner as the **NINODE** structure. When the file table overflows, the following warning message is displayed on the system console:

```
CONFIG: falloc - File table overflow (NFILE = number exceeded)
```

As a reminder, this parameter does not affect the number of open files per process (see the **NOFILES** parameter).

NMOUNT specifies how many mount table entries to allocate. Each entry represents a mounted filesystem. The *root* filesystem (/) is always the first entry. When full, the **mount**(S) system call returns the **EBUSY** error code. Because the mount table is searched linearly, this value should be as low as possible.

CMASK default mask used by **umask**(S) for file creation. By default, this is zero, meaning that the **umask** is not set in the kernel.

ETRUNC determines how the system handles attempts to use filenames that are longer than allowed for the particular filesystem architecture (either 14 or 255 characters). When **ETRUNC** is set to the default value of 0, attempts to create files with names greater than these respective limits fail with **ENAMETOOLONG**, which produces the "Filename too long" error message. This is the behavior mandated by POSIX FIPS requirements.

If **ETRUNC** is set to 1, filenames are silently truncated to either 14 or 255 characters depending on the filesystem architecture. Table 19-1 summarizes how the value of **ETRUNC** affects attempts to write filenames longer than supported on the filesystem:

Table 19-1 Filename truncation

Filesystem	ETRUNC 0	ETRUNC 1
S51K	Fails	Truncates to 14
XENIX	Fails	Truncates to 14
AFS	Fails	Truncates to 14
EAFS	Fails	Truncates to 255
ES51K	Fails	Truncates to 255

NOFILES specifies the maximum number of open files per process. Unless an application package recommends that **NOFILES** be changed, the default setting should be left as is. */bin/sh* uses three file table entries: standard input, standard output, and

standard error (0, 1, and 2 are normally reserved for stdin, stdout, and stderr, respectively). This leaves the value of **NOFILES** minus 3 as the number of other open files available per process. If a process requires up to three more than this number, then the standard files must be closed. This practice is not recommended and must be used with caution, if at all. If the configured value of **NOFILES** is greater than the maximum (11000) or less than the minimum (60), the configured value is set to the default (60), and a message is sent to the console. When the a process attempts to open more files than this limit, the following message is displayed on the terminal:

```
CONFIG: xsemfork - File table overflow (NOFILES = number exceeded)
```

SHLBMAX specifies the maximum number of shared libraries that can be attached to a process at one time. The maximum value is 16.

FLCKREC specifies the number of lock table entries, which determine the number of file regions that can be locked by the system. The "lock-sz" column on the **sar -v** report reports the number of entries being used in comparison to the number that are allocated. The proper value for this tuneable is determined by the applications being run on your system; if you are running applications that use file and record locking extensively, tune this to a higher value. If your applications use little file and record locking, the value of this tunable can be reduced.

NMPBUF number of Acer Fast File System cluster buffers. They are used to gather large disk requests before transferring the contents either to user space or to the buffer cache. **mpbufs** are large (16K each). The value **NMPBUF** should be 0 to 16, the latter being only appropriate only for a machine with 16 megabytes of memory or more and many users. When set to zero (the system default), the real value for this parameter is determined at boot time.

NMPHEADBUF
number of Acer Fast Filesystem standalone cluster buffer headers. This parameter must have a value from 0 to 600.

BFREEMIN number of buffers that must be on the freelist before a buffer is waited for. It should usually be 0. Machines that are used primarily for media copying, **uucp** transfers, and other applications that are both quasi-single-user and access lots of files see a performance boost by setting this to a value of around **NBUF**/10. The maximum value is 100.

S5CACHEENTS
number of name components in the filename cache. Must have a value of 1 to 1024. The recommended value for diverse workgroups is to make **S5CACHEENTS** large, roughly three times the value of **NINODE**. See "Tuning disk I/O parameters" (page 494) for more information about choosing values for **S5CACHEENTS**, **S5HASHQS**, and **S5OFBIAS**.

S5HASHQS number of hash queues for name cache. Must have a value of 1 to 1021. The recommended value for diverse workgroups is to make **S5HASHQS** a prime number roughly a quarter the size of **S5CACHEENTS**.

S5OFBIAS determines bias towards keeping open files around in cache. Must have a value of 0 to 256. A value of zero means that the names of open files have no special caching priority. If **S5OFBIAS** is equal to **S5CACHEENTS**, the names of open files almost always remain in the cache.

NGROUPS maximum number of supplemental groups. This value has a default of 8 and a maximum of 16.

HSNMOUNT maximum number of mounted High Sierra (CD-ROM) filesystems.

HSNINODE maximum number of open inodes for High Sierra (CD-ROM) filesystems.

HSNEXTENT maximum number of file extents for High Sierra (CD-ROM) filesystems.

DOSMOUNT maximum number of mounted DOS filesystems.

DOSINODE maximum number of open inodes for DOS filesystems.

Processes, memory management, and swapping

NPROC specifies how many process table entries to allocate. Each table entry represents an active process. The swapper is always the first entry, and */etc/init* is always the second entry. The number of entries depends on the number of terminal lines available and the number of processes spawned by each user. If the process table is filled up, users attempting to start new processes receive the following error message:

```
CONFIG: newproc - Process table overflow (NPROC = number exceeded)
```

The value of **NPROC** entry should be between 50 and 3,000. Use the **sar -v** report to see how many process table entries are being used compared to how many have been allocated. **NPROC** should be set to a value that is roughly 15% higher than the

largest value that shows in the *proc-sz* column of **sar -v**. Setting **NPROC** unnecessarily high wastes a lot of memory, especially since the proper value of the **NREGION** tunable parameter is determined relative to the value of **NPROC**.

MAXUP specifies how many concurrent user processes a non-super user is allowed to run. The entry is normally in the range of 15 to 60, with a maximum value of 300. This value should be at least 10% smaller than the value of **NPROC**. This value is per user identification number, not per terminal. For example, if 12 people are logged in on the same user identification, the default limit would be reached very quickly.

NREGION specifies how many region table entries to allocate. Most processes have three regions: text, data, and stack. Additional regions are needed for each shared memory segment and shared library (text and data) attached. However, the region table entry for the text of a "shared text" program is shared by all processes executing that program. Each shared-memory segment attached to one or more processes uses another region table entry.

NREGION can be set between 100 and 10,000. As a general rule, set **NREGION** to a value that is a bit more than 3 times the value of **NPROC**. When the region table overflows, the following message is displayed on the console:

```
CONFIG: allocreg - Region table overflow (NREGION = number exceeded)
```

MAXPMEM specifies the maximum number of physical pages. The default value of 0 specifies that all available physical memory be used.

ULIMIT specifies in 512-byte blocks the size of the largest file that an ordinary user can write. The default value is 2097151; that is, the largest file an ordinary user can write is one gigabyte. The super user can write a file as large as the filesystem can hold. The **ULIMIT** parameter does not apply to reads: any user can read a file of any size.

SPTMAP determines the size of the map entry array used for managing kernel virtual address space. Users should not modify this parameter.

The following six tunable parameters determine how often the **vhand** and **bdflush** daemons run and under what conditions. **vhand** is the paging daemon that is responsible for freeing up memory as the need arises. It uses a "least recently used" algorithm to approximate process working sets, and it writes out those pages that were not modified during some period of time to the disk. The page size is 4096 bytes. When memory is exceptionally tight, the working sets of entire processes may be swapped out. The default values of these parameters should be adequate for most applications.

AGEINTERVAL
specifies the number of clock ticks a process runs before its pages are aged.

GPGSLO specifies the low water mark of free memory in pages for **vhand** to start stealing pages from processes. Normally, **GPGSLO** is tuned to a value that is about 1/16 of pageable memory. Increase the value to make the daemon more active; decrease the value to make the daemon less active. This value must be a positive integer that is smaller than the value of **GPGSHI**.

GPGSHI specifies the high-water mark of free memory in pages for **vhand** to stop stealing pages from processes. Normally **GPGSHI** is set to a value that is about 1/10 of pageable memory. Increase the value to make the daemon more active; decrease the value to make the daemon less active. The value must be a positive integer that is greater than the value of **GPGSLO**, and is usually not greater than 10 megabytes.

GPGSMSK mask used by the paging daemon. The default is 0x00000420 (hex). Note that this appears as a decimal value in the system and should not be changed.

MAXSC specifies the maximum number of pages that are swapped out in a single operation. The default value is 8 and should not be changed.

MAXFC maximum number of pages that are added to the free list in a single operation. The default value is 1 and should not be changed.

MAXUMEM maximum size of a user's virtual address space in 4096-byte pages. This value cannot be greater than 131072.

MINARMEM minimum number of memory pages reserved for the text and data segments of user processes.

MINASMEM threshold value that specifies the number of memory and swap pages reserved for system purposes (unavailable for the text and data segments of user processes). If this limit is exceeded, the following message is displayed on the console:

```
CONFIG: swapdel - Total swap area too small (MINASMEM = number exceeded)
```

MINHIDUSTK
specifies the minimum data relocation value such that the user stack and data can share a page table. The **MINHIDUSTK** and **MINUSTGAP** values should not be changed.

MINUSTKGAP
See **MINHIDUSTK**.

MAXSLICE specifies in clock ticks the maximum time slice for user processes. After a process executes for its allocated time slice, that process is suspended. The operating system then dispatches the highest priority process and allocates to it **MAXSLICE** clock ticks. **MAXSLICE** must be a value from 25 to 100.

Clock

NCALL specifies how many call-out table entries to allocate. Each entry represents a function to be invoked at a later time by the clock handler portion of the kernel. This value must be in the range of 30 to 500. As a general rule of thumb, set **NCALL** to a value slightly higher than the number of enabled tty devices.

Software drivers may use call entries to check hardware device status. When the call-out table overflows, one of the following messages is displayed on the system console:

```
CONFIG: timeout - Timeout table overflow (NCALL = number exceeded)
CONFIG: MPX timeout - Timeout table overflow (NCALL = number exceeded)
```

It is difficult to determine exactly how many timeout table entries are required. Because each entry is relatively small, try doubling the value of **NCALL** after getting this message, which should be enough to solve the problem.

TIMEZONE specifies the **timezone** setting referred to in the **ctime**(S) system call. Note that the timezone value is a system default timezone and not the value of the **TZ** environment variable. This parameter can have a value from 0 to 1440.

DSTFLAG specifies the **dstflag** described for the **ctime**(S) system call. A value of 1 indicates Daylight Savings Time.

MultiScreens

The multiscreen parameters determine the number of console multiscreens that can run simultaneously on the system. Each multiscreen requires about 6K of memory, so if you need to save memory and are not using multiscreens heavily, set **NSCRN** to 4 and **SCRNMEM** to 24. When you do this, you must also disable multiscreens 5-12 or **getty** generates warning messages when the system goes to multiuser state. **NSCRN** and **SCRNMEM** can be set to even smaller values if you are absolutely sure of the number of multiscreens that will be used.

TBLNK controls the console screen saver feature on VGA consoles. It is the number of seconds before the screen blanks to save wear on the monitor. **TBLNK** can have a value of 0 to 32,767, with zero disabling screen blanks. The default is 0.

NSCRN number of multiscreens. A value of 0 configures this value at boot time based on the amount of memory installed. The maximum value is 12.

NSPTTYS number of pseudo-ttys on the system. The maximum is 256. This parameter can be altered using the **mkdev ptty** command, which also creates the additional device nodes. Note that this is not a multiscreen parameter but is displayed in this section of **configure**.

SCRNMEM number of 1024-byte blocks for console screen saves. A value of 0 configures this value at boot time based on the amount of memory installed. The maximum value is 128. Each multiscreen uses about 6K of memory, so when using a non-zero value for this parameter, make **SCRNMEM** = 6*NSCRN.

Message queues

The following tunable parameters are associated with interprocess communication messages:

MSGMAP specifies the number of entries in the messages memory map. Entries in the message map table say that ***n*** **MSGSSZ** segments of memory are free at ***x*** address. This number really measures how fragmented you expect your map to get. It can be small if you always send a few large messages, and should be large if you send a lot of small messages. The safest number is roughly half of **MSGSEG**. The default value is 100. Each entry contains 8 bytes.

MSGMAX maximum size of a message. The default value is 2,048. Although the maximum possible size the kernel can process is 64 kilobytes -1, the limit is 8,192.

MSGMNB maximum number of bytes that all the messages in any one message queue can occupy. The default value is 4,096.

MSGMNI maximum number of message queues system-wide (id structure). The default value is 50.

MSGTQL number of messages that can be stored by the kernel; that is, the maximum number of unread messages at any given time. The default value is 40. Each entry contains 12 bytes.

MSGSSZ unit of memory allocation for storing a message. If **MSGSSZ** is 8, then ***n*** 8-byte segments of memory are used for storing a message. An 18-byte message would therefore require three message segments to be stored in a message queue. The default value is 8.

MSGTQL

MSGSEG number of **MSGSSZ** segments of memory are allocated at kernel startup for holding messages. Therefore, **MSGSEG*MSGSSZ** bytes of memory are allocated for messages. If **MSGSEG** is set at 0, then the kernel will auto-configure based on the total amount of system memory. If **MSGSEG** is 0, then **MSGMAX** and **MSGMNB** are both set to **MSGSEG*MSGSSZ**.

> **NOTE** The **IPC_NOWAIT** flag can be passed into many of the **msg** system calls. If this flag is passed, then the system calls will fail immediately if there is no space for a message. If this flag is not passed, then the system calls will sleep until there is room for the message.

To determine adequate values for each of the parameters, compute the maximum size and number of messages desired, and allocate that amount of space. For example, if the system will have at most 80 messages of 1 kilobyte each pending, then **MSGTQL** should be set to 80, and **MSGSEG** is computed as:

- 80 messages of 1K each = 80 Kbytes total message space
- Divide total message space by **MSGSSZ** to get **MSGSEG**
- If **MSGSSZ**=8, then **MSGSEG**=80K/8=10K=10240

Semaphores

The following tunable parameters are associated with interprocess communication semaphores:

SEMMAP size of the control map used to manage semaphore sets. The default value is 10. Each entry contains 8 bytes.

SEMMNI number of semaphore identifiers in the kernel. This is the number of unique semaphore sets that can be active at any given time. The default value is 10. Each entry contains 32 bytes.

SEMMNU number of undo structures in the system. The default value is 30. The size is equal to 8*(**SEMUME** + 2) bytes.

SEMMSL maximum number of semaphores per semaphore identifier. The default value is 25.

SEMOPM maximum number of semaphore operations that can be executed per **semop** system call. The default value is 10. Each entry contains 8 bytes.

SEMUME maximum number of undo entries per undo structure. The default value is 10. The size is equal to 8***SEMMNU** bytes.

SEMVMX maximum value a semaphore can have. The default value is 32,767, which is the maximum value for this parameter.

SEMAEM adjustment on exit for maximum value, alias **semadj**. This value is used when a semaphore value becomes greater than or equal to the absolute value of **semop**, unless the program has set its own value. The default (and maximum) value is 16,384.

SEMMNS number of semaphores in the system. The default value is 60. Each entry contains 8 bytes.

XSEMMAX size of the XENIX semaphore table that determines the maximum number of XENIX special semaphores allowed systemwide. The minimum value for **XSEMMAX** is 20, the maximum value is 60, and the default value is 60. When the XENIX semaphore table overflows, the following message is displayed on the console:

```
CONFIG: xsem_alloc - XENIX semaphore table overflow (XSEMMAX = number exceeded)
```

Shared data

The following tunable parameters are associated with interprocess communication shared memory:

SHMMAX maximum shared-memory segment size. The default value is 524,288.

SHMMIN minimum shared-memory segment size. The default value is 1.

SHMMNI maximum number of shared-memory identifiers systemwide. The default value is 100. Each entry contains 52 bytes.

SHMSEG number of attached shared-memory segments per process. The default value is 6. The maximum value is 15.

SHMALL maximum number of in-use shared-memory text segments. The default value is 512.

XSDSEGS maximum number of XENIX special shared-data segments allowed system wide. The minimum value for **XSDSEGS** is 1, the maximum value is 150, and the default value is 25. When the XENIX shared data table overflows, the following message is displayed on the console:

```
CONFIG: xsd_alloc - XENIX shared data table overflow (XSDSEGS = number exceeded)
```

XSDSLOTS number of slots per XENIX shared data segment. The maximum number of XENIX special shared data segment attachments system wide is **XSDSEGS*XSDSLOTS**. The minimum value for **XSDSLOTS** is 1, the maximum value is 10, and the default value is 3.

System name

NODE system name. The value of **NODE** must not be greater than eight characters.

STREAMS data

STREAMS is a facility for UNIX system communication services. It supports the implementation of services ranging from complete networking protocol suites (such as TCP/IP) to individual device drivers. STREAMS defines standard interfaces for character I/O. The associated mechanism is simple and open-ended, consisting of a set of system calls, kernel resource and kernel routines.

STREAMS use system resources that are limited by values defined in kernel configuration modules. Depending on the demand that you and other system users place on these resources, your system could run out of STREAMS resources if you do not first reset the allocations in the kernel configuration modules. If you do not use STREAMS-based applications heavily and are trying to reduce the size of the kernel, reducing the values of these parameters (especially those that control the STREAMS data blocks) can free up a significant amount of memory.

Running out of some STREAMS resources (such as those controlled by the **NQUEUE** and **NSTREAM** parameters) generate kernel error messages. Running out of data blocks (controlled by the **NBLK*n*** parameters) does not generate error messages, because the system will just use a larger buffer. However, running out of buffers of a certain size can cause performance problems, or may cause MultiScreens to terminate abnormally, or cause the system to lock or hang.

STREAMS parameters can be checked using the **strstat** function of the **crash**(ADM) utility. Before changing the STREAMS parameters **NQUEUE**, **NSTREAM**, **NUMSP**, or **NBLK*n***, check the **crash strstat** output. See "crash strstat (statistics about STREAMS buffers)" (page 506) for more information.

The following tunable parameters are associated with STREAMS processing.

NQUEUE number of STREAMS queues to be configured. Queues are always allocated in pairs, so this number should be even. A minimal STREAM contains four queues (two for the STREAM head, two for the driver). Each module pushed on a STREAM requires an additional two queues. A typical configuration value is 4***NSTREAM**. The following message is displayed on the console if you exceed the number of STREAMS queues:

```
CONFIG: id - out of STREAMS queues (NQUEUE = number exceeded)
```

NSTREAM number of STREAM-head (**stdata**) structures to be configured. One is needed for each STREAM opened, including both STREAMs currently open from user processes and STREAMs linked under multiplexers. The recommended configuration value is highly application-dependent, but a value of 32 to 40 usually suffices on a computer for running a single transport provider with moderate traffic. When the number of STREAM head structures is exceeded, one of the following messages is displayed on the console:

```
CONFIG: stropen1 - Out of streams (NSTREAM = number exceeded)
CONFIG: stropen2 - Out of streams (NSTREAM = number exceeded)
```

NBLK*n* **NBLK4** through **NBLK4096** control the number of STREAMS data blocks and buffers to be allocated for each size class. Message block headers are also allocated based on these numbers: the number of message blocks is 1.25 times the total of all data block allocations. This provides a message block for each data block, plus some extras for duplicating messages (kernel functions **dupb**() and **dupmsg**()). The optimal configuration depends on both the amount of primary memory available and the intended application. When the system cannot allocate STREAMS blocks, the following message is displayed on the console:

```
CONFIG: sp: spclose - Cannot allocate STREAMS block (NBLCKnnnn = number exceeded)
```

NMUXLINK maximum number of multiplexer links to be configured. One link structure is required for each active multiplexer link (**STREAMS I_LINK ioctl**). This number is application-dependent; the default allocation of 87 guarantees availability of links.

NSTRPUSH maximum number of modules that may be pushed onto a STREAM. This prevents an errant user process from consuming all of the available queues on a single STREAM. By default this value is 9, but in practice, existing applications have pushed at most four modules on a STREAM.

NSTREVENT initial number of STREAM event cells to be configured. STREAM event cells are used for recording process-specific information in the **poll** system call. They are also used in the implementation of the STREAMS **I_SETSIG ioctl** and in the kernel **bufcall**() mechanism. A rough minimum value to configure would be the expected number of processes to be simultaneously using **poll** times the expected number of STREAMS being polled per process, plus the expected number of processes expected to be using STREAMS concurrently. The default is 256. Note that this number is not necessarily a hard upper limit on the number of event cells that are available on the system (see **MAXSEPGCNT**).

MAXSEPGCNT
number of additional 4096-byte pages of memory that can be dynamically allocated for event cells. If this value is 0, only the allocation defined by **NSTREVENT** is available for use. If the value is not 0 and if the kernel runs out of event cells, it will under some circumstances attempt to allocate an extra page of memory from which new event cells can be created. **MAXSEPGCNT** places a limit on the number of pages that can be allocated for this purpose. Once a page is allocated for event cells, however, it cannot be recovered later for use elsewhere. The default value is 1.

STRMSGSZ maximum allowable size of the data portion of any STREAMS message. This should usually be set just large enough to accommodate the maximum packet size restrictions of the configured STREAMS modules. If it is larger than necessary, a single **write** or **putmsg** can consume an inordinate number of message blocks. The default value of 4,096 is sufficient for existing applications.

STRCTLSZ maximum allowable size of the control portion of any STREAMS message. The control portion of a **putmsg** message is not subject to the constraints of the minimum/maximum packet size, so the value entered here is the only way of providing a limit for the control part of a message. The default value of 1,024 is more than sufficient for existing applications.

STRLOFRAC percentage of data blocks of a given class at which low-priority block allocation requests are automatically failed. For example, if **STRLOFRAC** is 40 and there are forty-eight 256-byte blocks, a low-priority allocation request fails when more than nineteen 256-byte blocks are already allocated. The parameter helps prevent deadlock situations by starving out low-priority activity. The default value of 80 works well for most applications. **STRLOFRAC** must be greater than or equal to 0 and less than or equal to **STRMEDFRAC**. For slightly more efficient use of STREAMS buffers, increase the value of **STRLOFRAC** to around 95 and increase **STRMEDFRAC** as discussed next.

STRMEDFRAC
percentage cutoff at which medium priority block allocations are failed (see **STRLOFRAC**). The default value of 90 works well for most applications. For slightly more efficient use of STREAMS buffers, increase the value of **STRMEDFRAC** to around 98 and increase **STRLOFRAC** as discussed above. **STRMEDFRAC** must be greater than or equal to **STRLOFRAC** and less than or equal to 100. (There is no cutoff fraction for high-priority allocation requests; it is effectively 100.)

NLOG number of minor devices to be configured for the log driver; the active minor devices are 0 through (**NLOG**–1). The recommended value of 3 services an error logger (**strerr**) and a trace command (**strace**), with one left over for miscellaneous usage. If only an error logger and a tracer are to be supported, this number can be set to 2. If there are several daemons for an application that may be submitting log messages, this number can be increased to accommodate the extra users.

NUMSP determines the number of STREAMS pipe devices (*/dev/sp*) supported by the system. Users should not modify this parameter.

NUMTIM maximum number of STREAMS modules that can be pushed by the Transport Library Interface (TLI). This value controls the number of data structures used to hold pushed STREAMS modules configuration data. Users should not modify this parameter.

NUMTRW number of Transport Library Interface (TLI) read/write data structures to allocate in kernel data space. Users should not modify this parameter.

Event queues and devices

EVQUEUES maximum number of open event queues systemwide. The acceptable range is 1-16, with a default value of 8.

EVDEVS size of the event table that determines the maximum number of devices attached to event queues systemwide. The acceptable range is 1-16, with a default value of 16. When the event table overflows, the following message is displayed on the console:

```
CONFIG: event - Event table full (EVDEVS = number exceeded)
```

EVDEVSPERQ

maximum number of devices per event queue. The acceptable range is 1-16, with a default value of 3. When the event channel overflows, the following message is displayed on the console:

```
CONFIG: event - Event channel full (EVDEVSPERQ = number exceeded)
```

Hardware dependent parameters

DMAEXCL specifies whether simultaneous DMA requests are allowed. Some computers have DMA chips that malfunction when more than one allocated channel is used simultaneously. **DMAEXCL** is set to 0 by default to allow simultaneous DMA on multiple channels. If this causes a problem, set it to 1.

KBTYPE determines the logical character protocol used between the keyboard and the keyboard driver. This tunable is usually set to 0; a value of 1 specifies the AT logical character protocol which is recognized by the console driver but not by the X server. All AT-compatible keyboards support both modes.

VGA_PLASMA
is set to 1 if a VGA gas plasma display is present, 0 if not.

NSHINTR maximum number of devices sharing the same interrupt vector. This has a default value of 8. Users should not modify this parameter.

DO387CR3 controls the setting of high-order bits of Control Register 3 (CR3) when an 80387 math coprocessor is installed. Because of design defects in Intel's 80387 chip (B1 stepping,) the Intel 80387 math coprocessor may not operate correctly in some computers. The problem causes CPU to hang when DMA/paging/coprocessor accesses occur. You can work around this problem by changing the **DO387CR3** parameter from the default value of 0 to 1. The default value is 0 (switched off).

MODE_SELECT
determines whether mode-select checking is performed on parallel ports. The default value of 1 enables mode-select checking. Setting this value to 0 disables this feature, and removes the printing delay incurred by these checks.

WARNING The following two parameters are not available via **configure**(ADM). Use the **idtune**(ADM) command instead. Never edit the *mtune*(F) file.

NAHACCB number of mailboxes available for the Adaptec driver to talk to the Adaptec hardware. The higher the number, the less likely it is that the driver has to "sleep" before "talking" to the hardware. It is not normally necessary to modify this parameter.

SDSKOUT number of outstanding requests on a SCSI disk at any one time.

Security

The security default (High, Improved, Low, or Traditional) can be selected as discussed in the chapter on maintaining system security in this guide; the security parameters can be set to modify the behavior of the security features and to ensure compatibility with utilities that expect traditional UNIX system behavior. Each of these parameters can be set to 0 (off) or 1 (on).

SECLUID controls the enforcement of LUID (login user ID). Under C2 requirements, every process must have a login user ID (LUID). This means that processes that 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 default is set to a lesser default (that is, not "High"), enforcement of login user ID (LUID) is relaxed and **setuid** programs do not require an LUID to run.

SECSTOPIO controls whether the kernel implements the **stopio**(S) system call. When **SECSTOPIO** is set to 1, the kernel acts on **stopio**(S) calls; when it is set to 0, the kernel ignores **stopio** calls. The **stopio** system 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 may be killed.

stopio(S) is used by **initcond**(ADM), which is called by **getty**(M) immediately before starting user interaction and by **init**(M) immediately after an interactive session has terminated.

SECCLEARID

controls the clearing of SUID/SGID bits when a file is written. Under C2 requirements, the set user ID (SUID or **setuid**) and set group ID (SGID 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. 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.

The following table summarizes the initial settings of the security parameters for each security default.

Parameter	Low	Traditional	Improved	High
SECLUID	off	off	off	on
SECSTOPIO	off	on	on	on
SECCLEARID	off	on	on	on

Asynchronous I/O

NAIOPROC size of the AIO process table that determines the number of processes that may be simultaneously performing asynchronous I/O. The default value is 5. When the AIO process table overflows, the following message is displayed on the console:

```
CONFIG: aio_memlock - AIO process table overflow (NAIOPROC = number exceeded)
```

NAIOREQ size of the AIO request table that determines the maximum number of pending asynchronous I/O requests. The default value is 120. When the AIO request table overflows, the following message is displayed on the console:

```
CONFIG: aio_breakup - AIO request table overflow (NAIOREQ = number exceeded)
```

NAIOBUF size of the AIO buffer table that determines number of asynchronous I/O buffers. This should always be set to the same value as **NAIOREQ**. When the AIO buffer table overflows, the following message is displayed on the console:

```
CONFIG: aio_breakup - AIO buffer table overflow (NAIOBUF = number exceeded)
```

NAIOHBUF number of internal asynchronous hash queues.

NAIOREQPP maximum number of asynchronous I/O requests that a single process can have pending. The default value is 120, meaning that a single process can potentially exhaust all asynchronous I/O resources.

NAIOLOCKTBL
number of entries in the internal kernel table for asynchronous I/O lock permissions. The default value is 10. If there are many entries in the */usr/lib/aiomemlock* file, this value may need to be increased. When the AIO lock table overflows, the following message is displayed on the console:

```
CONFIG: aio_setlockauth - AIO lock table overflow (NAIOLOCKTBL = number exceeded)
```

Boot load extension parameters

NOTE This group of parameters is not available via **configure**(ADM). Use the **idtune**(C) command instead. Never edit the *mtune*(F) file.

EXTRA_NDEV
number of extra device slots in **fmodsw[]**, **io_init[]**, and **io...[]**. It defines the number of slots reserved in the device driver tables for Boot Time Loadable Drivers.

EXTRA_NEVENT
number of extra event slots. It defines the number of slots reserved in the event driver tables for Boot Time Loadable Drivers.

EXTRA_NFILSYS
number of extra types of filesystem. It defines the number of extra types of filesystem that can be mounted at boot time.

MAX_BDEV maximum number of block devices (**bdevcnt** is at least this value). It defines the minimum number of entries in **bdevsw[]**, the block device switch table.

MAX_CDEV maximum number of character devices (**cdevcnt** is at least this value). It defines the minimum number of entries in **cdevsw[]**, the character device switch table.

Multiprocessing parameters

MAXACPUS number of additional CPUs. Do not modify this parameter.

Chapter 20

Troubleshooting reference

This chapter contains notes and tips about specific problems that may arise. These notes are organized alphabetically, grouped by related topic when appropriate (for example, security system troubleshooting and tape drive troubleshooting). Reading over this material leisurely before you have a problem may help you troubleshoot more quickly. If you are looking for information about a specific situation, you may find the index is the easiest way to access the specific piece of information you need.

Boot troubleshooting

This section discusses reasons why a system that has booted successfully in the past may not boot now. These situations usually occur as a result of a power failure or system panic that corrupts the *root* filesystem, although configuration changes, hardware failure and human error can also cause these situations.

The *Installation Guide* gives instructions for troubleshooting a system that does not boot during installation.

If you cannot boot your system, first check the following:

- Is the system plugged in?
- Are any cables loose, disconnected, or misconnected?
- Are the cable chains terminated properly?
- Does the floppy drive contain a floppy that is not a boot floppy?
- Has your hard disk developed a bad track? A bad track on the disk may corrupt system files that are required for booting the system. See "Mapping a bad track" (page 562) for information on how to recover from this situation.

Restoring missing or corrupted system files

On rare occasions, one or more of the critical system files is accidentally modified or removed, preventing the system from booting or operating correctly. In cases where your system does not boot, you must boot from floppy disks in order to access the system so that you can restore the critical files from backups.

To boot and access a system that does not boot from the hard disk, you must have made an Emergency Boot Floppy set as directed in the *Installation Guide*. This set consists of the boot floppy and the *root* filesystem floppy. The boot floppy contains three files necessary for booting and loading the UNIX system kernel: */boot*, */etc/default/boot*, and */unix*. The *root* filesystem floppy contains a subset of the UNIX system utilities that you can use to restore your system.

NOTE We recommend that you have a separate Emergency Boot Floppy set for each system or further corruption can result. Systems that have *identical* hardware and software configurations can share an Emergency Boot Floppy set.

If you have not made these floppies, you must reinstall the operating system. In some cases, if you do not also have a backup of the *root* filesystem, you must reinstall the operating system. To do this, follow the instructions in the *Installation Guide*.

The following sections discuss common reasons your system will not boot and give instructions for recovering. These conditions are listed in the order they will probably occur.

/boot not found

If your system displays the following message when you turn on the power to your computer, the */boot* file is missing:

```
/boot not found
Stage 1 boot failure: error loading /boot
```

The */boot* file contains the **boot**(HW) program, which loads and executes the kernel each time you turn on the computer.

If */boot* is missing, use the following procedure to boot the system from the Emergency Boot Floppy set so that you can then restore the file:

1. Insert the boot floppy in the drive and reboot the machine. This executes the initial boot from the boot floppy.

2. At the Boot: prompt, enter:

   hd(40)unix

   After you boot the system from the floppy, this command loads the kernel from the hard disk.

3. Bring up the system in single-user mode by entering the *root* password at the prompt.

4. Mount the boot floppy from the command line:

   mount -r /dev/fd0 /mnt

   The floppy must *not* be write protected.

5. While the floppy is in the drive, restore the */boot* file by entering the following command at the system prompt:

   cp /mnt/boot /

   This places a new copy of the */boot* file on the hard disk.

6. Before you remove the floppy disk from the drive, unmount the boot floppy (*/dev/fd0*) by entering:

   umount /mnt

7. Remove the floppy disk from the drive and bring down the system using **haltsys**(ADM).

8. Reboot the system from the hard disk by pressing ⟨Return⟩ at the Boot: prompt.

unix not found

If the system displays the "unix not found" message when you turn on the power to your computer, the */unix* file is missing. The */unix* file contains the bootable image of the UNIX system kernel. If */unix* is missing, you can boot from another kernel file, such as */unix.old*, */etc/conf/cf.d/unix*, or */etc/conf/cf.d/unix.old* by specifying the complete pathname of the file at the Boot: prompt.

If there are no other bootable kernel image files on the system, use the following procedure to boot the system from the Emergency Boot Floppy set so that you can restore the */unix* file:

1. Insert the boot floppy in the drive and reboot the machine.

2. At the Boot: prompt, enter:

   fd(64)unix root=hd(40) swap=hd(41)

   This loads the kernel from the boot floppy and mounts the *root* filesystem on the hard disk.

NOTE If you have changed the location of the swap device, you should substitute it for **hd(41)**.

3. Bring up the system in single-user mode.
4. Mount the boot floppy from the command line using:

   mount -r /dev/fd0 /mnt

5. While the floppy is in the drive, restore */unix* with the following command:

   cp /mnt/unix /

   This copies the */unix* kernel file from the boot floppy to the hard disk.

6. Before you remove the floppy disk from the drive, unmount the floppy filesystem (*/dev/fd0*) by entering:

   umount /mnt

7. Remove the floppy from the drive and bring down the system with the **haltsys**(ADM) command.
8. Reboot the system by pressing ⟨Return⟩ at the Boot: prompt.
9. Relink the kernel to ensure that the correct devices are linked in.

Cannot load floating point emulator

If your computer does not have a 387 math coprocessor chip and the */etc/emulator* file is missing or corrupted, the boot fails with one of the following messages:

```
WARNING:  Cannot load floating point emulator
WARNING:  Zero length FP emulator file
WARNING:  Cannot read emulator file header
```

(If the 387 chip is present, the kernel recognizes it in the hardware recognition boot message.)

If the boot fails with this message, use the following procedure to boot the system and restore */etc/emulator*:

1. Insert the boot floppy in the drive and reboot the machine.
2. At the Boot: prompt, press ⟨Return⟩ and when instructed, insert the *root* floppy. This boots the system and mounts the *root* filesystem from the floppies.
3. You should clean the *root* filesystem next:

   /bin/fsck -s /dev/hd0root

4. Working from the floppy disk, use the following **mount**(ADM) command to mount the hard disk *root* filesystem to */mnt*:

   mount /dev/hd0root /mnt

   If **mount** fails, refer to "Filesystem mount failed" (page 574) for information on checking the hard disk with **fsck**(ADM).

5. Copy */etc/emulator* from the *root* filesystem on the floppy to the mounted hard disk:

   cp /etc/emulator /mnt/etc/emulator

6. Unmount the hard disk:

   umount /mnt

7. Make sure that the floppy is still in the drive and reboot the system with the **haltsys**(ADM) command.
8. When you see the "Press any key" prompt, remove the floppy from the drive. Press ⟨Return⟩ at the Boot: prompt to boot from the hard disk.

System hangs at login prompt

If the system boots correctly but hangs at the login prompt after you enter multiuser mode, try other multiscreens or serial terminals.

NOTE It is also possible that the */etc/inittab* file has been corrupted. See "System cannot enter multiuser mode" (page 556) for more information.

If you cannot get a response from any ttys, the */bin/login* file may be missing. The */bin/login* file contains the **login**(M) program. This command is run at the beginning of each terminal session to allow users access to the system. Follow these steps to restore */bin/login*:

1. Power-cycle the machine and press ⟨Return⟩ at the Boot: prompt.
2. At the prompt, enter the *root* password to go into single-user mode.
3. Make the following selections from **custom**:

   Install
   SCO UNIX System V Operating System
   Service Components
   SCO UNIX System V Runtime System
   Files
   UNIX Run Time System
   ⟨F5⟩
   login

4. Exit **custom** when the file is restored.

System cannot enter multiuser mode

If the system fails to enter multiuser mode when you press ⟨Ctrl⟩**d**, or the following message is displayed at boot time, the */etc/inittab* file is missing:

```
INIT: Cannot open /etc/inittab errno: 2
INIT: SINGLE USER MODE
```

The */etc/inittab* file contains instructions for **init**. When *inittab* is missing, **init** cannot execute the system start-up instructions and the system cannot enter multiuser mode. When you press ⟨Ctrl⟩**d**, the system remains in single-user mode and displays the error message above.

As */etc/inittab* contains instructions that reference other parts of the system, a special */etc/inittab* is written to the *root* filesystem floppy when you create the Emergency Boot Floppy set. Because of this, the procedure for restoring your system is more complex than simply copying */etc/inittab* from the floppy filesystem to the hard disk.

To restore *inittab*, you must recreate the kernel environment. To do this, use the following procedure:

1. Go into system maintenance mode.
2. Enter the following commands:

   cd /etc/conf/cf.d
   touch /etc/.new_unix
   ../bin/idmkenv

3. You see the following:

```
The kernel environment includes device node files and /etc/inittab.
The new kernel may require changes to /etc/inittab or device nodes.
Do you want the kernel environment rebuilt? (y/n)
```

   Enter " y " and press ⟨Return⟩.

4. The following is displayed:

```
The kernel has been successfully linked and installed.
        To activate it, reboot your system.
Setting up kernel environment
```

5. Enter the **haltsys** command to reboot your system:

   haltsys

When the system reboots, the new */etc/inittab* file is in place.

execlp of /bin/sulogin failed

If the system displays the "INIT: execlp of /bin/sulogin failed; errno = 2" error message and goes directly into multiuser mode (run-level 2) at boot time, the */bin/sulogin* file is missing from the hard disk. The **sulogin**(ADM) utility must be present on the system to access single-user mode. If this file is missing, log in as *root* and make the following selections from **custom**:

Install
SCO UNIX System V Operating System
Service Components
SCO UNIX System V Runtime System
Files
UNIX Run Time System
⟨F5⟩
sulogin

Exit **custom** when the file is restored.

/etc/bcheckrc not found

If the system displays the "INITSH: /etc/bcheckrc: not found" error message when you boot up, the */etc/bcheckrc* file is missing. The **init** utility executes **bcheckrc** according to instructions in the */etc/inittab* file whenever the system is booted. This utility checks the *root* filesystem and repairs it, if necessary. The */etc/bcheckrc* file should be on the hard disk when you boot the system.

If */etc/bcheckrc* is missing, use the following procedure to recover it:

1. Bring up the system in single-user mode.
2. Clean the *root* filesystem manually with **fsck** before doing anything on the system:

   /etc/fsck -b /dev/root

3. When filesystem has been checked and, if necessary, repaired, make the following selections from **custom**:

   Install
   SCO UNIX System V Operating System
   Service Components
   SCO UNIX System V Runtime System
   Files
   UNIX Run Time System
   ⟨F5⟩
   bcheckrc
   ⟨Return⟩

   Exit **custom** when the file is restored.

4. Enter **haltsys** at the prompt and reboot the system.

For more information on **bcheckrc**, see the **brc**(ADM) manual page.

System hangs at boot time

If the boot process hangs after the "Kernel: i/o bufs" message, the */etc/init* file is missing from the system. The */etc/init* file contains the **init**(M) program. Once started, the **init** process spawns all other processes on the system, so if it is missing, no new processes are started.

Use the following procedure to restore */etc/init*:

1. Insert the boot floppy from the Emergency Boot Floppy Set in the floppy drive and reboot the machine.
2. Press ⟨Return⟩ at the Boot: prompt and when instructed, insert the *root* floppy.
3. Mount the hard disk *root* filesystem:

   mount /dev/hd0root /mnt

   If **mount** fails, check the hard disk with the **fsck**(ADM) command as discussed in "Filesystem mount failed" (page 574).
4. Copy the */etc/init* file from the *root* filesystem on the floppy to the mounted hard disk:

   cp /etc/init /mnt/etc/init
5. Unmount the hard disk by entering **umount /mnt**.
6. With the floppy in the drive, reboot the system with the **haltsys**(ADM) command.
7. Remove the floppy from the drive when you see the "Hit any key" prompt. Press ⟨Return⟩ at the Boot: prompt to boot from the hard disk.

Console troubleshooting

This section covers some common problems that you may have with the console on your system:

- The keyboard on your console locks up.
- How to prevent keyboard lockup by applying a special "patch".
- The console does not recognize keyboard input because you selected the wrong console keyboard type.
- You cannot log into the console multiscreens when the system is in multiuser mode.

Console keyboard locks up

When the system does not respond to input from the console keyboard, the situation is known as "keyboard lockup." Console keyboard lockup only affects keyboards that are attached to the computer's console, not standard terminals that are attached to serial lines.

You may be experiencing keyboard lockup if the following statements are true:

- The system console keyboard cannot be used to enter data or perform any tasks.
- You cannot use the ⟨Alt⟩⟨F1⟩ through ⟨F12⟩ keys to switch multiscreens, and the ⟨CapsLock⟩ key does not turn the CapsLock light on or off.
- Other terminals on the system continue to work.
- Printers or other devices continue to work, and the system is still running.

Before trying to fix a locked keyboard, make sure that:

- You did not accidentally press ⟨Ctrl⟩**s** (which stops the screen from scrolling). To check this, press ⟨Ctrl⟩**q** and then see if you can enter characters from the keyboard.
- If your computer has a Keyboard Lock key, it is not in the locked position.
- The keyboard is plugged into the correct socket.
- The system itself is still running.

Check a terminal to see if it is still working and that you can perform system tasks, such as logging in and checking the date. If you do not have a terminal, watch the hard disk access light, if your computer has one.

NOTE You cannot use other terminals, and the hard disk access light may not flash if you are in single-user mode.

If it flashes periodically, at least once every 30 seconds, the system is still running and is using the hard disk.

If the console keyboard is still locked after checking these suggestions, you can try unplugging the console keyboard and plugging it in again.

WARNING Unplugging the keyboard and reconnecting it while the system is powered up can damage some computers.

If this fixes the problem, your situation is definitely keyboard lockup. If this last step does not fix the problem, you may still have keyboard lockup.

Preventing console keyboard lockup

You can prevent keyboard lockup by applying a special "patch" that changes the operating system kernel. (The kernel is the main program of the operating system that is always running in memory.)

NOTE This patch disables the keyboard lights, so you should use it only if you have tried the other approaches.

The procedure for using the patch is as follows:

1. Get the system console working, if it is not. Reboot the system if you have to and bring it up in single-user mode.

   If you did not reboot, log in as *root* on the system console and shut the system down to single-user mode with the **shutdown** command:

   /etc/shutdown su

   (For more information, see the **shutdown**(ADM) manual page.)

2. Once the system is in single-user mode, back up the kernel with the following commands:

   cd /
   cp unix unix.00

3. Now, patch the kernel with the following commands:

   /etc/_fst -w /unix
   ledspresent/w 0
   $q

4. Shut down the system using **/etc/shutdown**.

5. When you see the "Normal System Shutdown" message, press any key to reboot the system. You have now fixed the keyboard lockup problem.

You can apply this patch permanently by entering the following commands:

```
cd /etc/conf/pack.d/cn
copy -om Driver.o Driver.o.save
/etc/_fst -w Driver.o
ledspresent/w 0
$q
```

Wrong console keyboard type

If your console keyboard is an XT or other non-AT keyboard and the operating system is configured for use with an AT keyboard, the system does not recognize input from the keyboard. For information on testing and switching keyboard modes, see the section on console keyboard type selection in the "Using the system console and color displays" appendix in this guide.

Cannot log into console

If you try to log into the console in multiuser mode, and the system displays the following error message:

```
Cannot obtain database information on this terminal
```

Refer to "Cannot obtain database information on this terminal" (page 598) for more information.

cron, at, and batch troubleshooting

cron(C) is the clock daemon that schedules jobs to run automatically at a set time. The **at** and **batch** commands can be used by users to run non-interactive jobs once. See the **at**(C) manual page for more information.

cron daemon is not running

The **cron** daemon executes commands submitted with the **at**(C), **batch**(C), and **crontab**(C) commands at specified dates and times. (The **cron** daemon is started automatically by a script in the */etc/rc2.d* directory when the system is started in multiuser mode.) If **cron** is not running and you try to submit a job with one of these programs, the system displays the following message:

```
cron may not be running - call your system administrator
```

Another indicator that **cron** is not running is if previously scheduled jobs are not being executed. Use the following command to see if the **cron** daemon is running:

ps -ef | grep cron

If there is no **cron** process, then the daemon is not running. To start **cron**, log in as *root* and enter the following command:

sd cron

If this does not start the **cron** process, simply reboot the system. During startup, the system displays a message like the following:

```
! *** cron started *** pid = 140 Tue Apr 21 14:02:47 PDT 1992
```

The jobs scheduled with **at**, **batch**, and **crontab** should now execute properly. If **at** and **batch** continue to report error messages, refer to "at command fails: cannot change mode of job" (page 562) for more information.

NOTE If this is a persistent problem, you can start **cron** logging by entering **CRONLOG=YES** in the */etc/default/cron* file. After **cron** is started, the */usr/spool/cron/log* file will hold a record of all **cron** transactions; this information may help you determine why **cron** is halting.

at command fails: cannot change mode of job

If you do not have the kernel authorization, **chmodsugid**, the system displays the following message when you try to invoke **at**(C) or **batch**(C):

```
can't change mode of job
```

To grant **chmodsugid** authorization so that the user can use **at**, use the following procedure:

1. Log in as *root*.
2. Invoke **sysadmsh** and select:

   Accounts ⇨ User ⇨ Examine:Privileges
3. Add "chmodsugid" to the Kernel Authorization field.

To allow the changes to take effect, the user must log out and then log in again.

This is different from the situation in which the user is not *authorized* to use **at**. If a user who is not authorized tries to use **at**, the system displays:

```
at: you are not authorized to use at.  Sorry.
```

To allow a user to use **at**, see "Allowing or denying individuals the use of cron" (page 630) for more information. Use the **sysadmsh**(ADM) selection Jobs ⇨ Authorize ⇨ Scheduled ⇨ User.

Disk problems

This section covers some common problems that you may have with the hard disk on your system:

- bad tracks
- disk load balancing

Mapping a bad track

The hard disk sometimes develops a bad track, which is an area of the disk that is not reliable for data storage. The hard disks supported on SCO systems allow you to map the bad track into the bad track table that lists any areas of the hard disk that should not be used. The bad tracks listed on the table are aliased to good tracks so that the operating system avoids the areas of the disk that cannot be read or written.

If your hard disk develops a bad track after the system is installed and running, an error message like the following is displayed on the console:

```
wd: ERROR on fixed disk ctlr=0 dev=0/47 block=31434 cmd=00000020
    status=00005180, sector = 62899, cylinder/head = 483/4
```

If this error occurs, use **badtrk**(ADM) to create a new bad track table so that the system can avoid the new bad track(s). The **badtrk** utility is a menu-driven utility for viewing, adding, or deleting entries in the bad track table.

WARNING SCSI disks do not use **badtrk**. Use the **scsibadblk**(ADM) utility to map any bad blocks. If your adapter supports Automatic Write Remapping (AWRE), you can use **scsibadblk** to enable this feature.

To use **badtrk**, do the following:

1. Reboot the system and enter single-user mode. Do not use **shutdown su**.
2. Enter **badtrk** at the system prompt.
3. Select option 2 to scan the disk, then select option 1 to scan the entire UNIX system partition.
4. Indicate whether you want to do a quick or thorough scan and, at the destructive scan prompt, enter "n".

   NOTE Run **badtrk** in nondestructive mode to save the data on your hard disk. The thorough scan is recommended if new bad tracks have appeared.

5. When the scan is complete, the main menu reappears. Select "q" to return to the system.

The **badtrk** utility automatically enters any flaws it detects in the bad track table (as long as there is room in the table). For more information on the options to **badtrk**, see the manual page on **badtrk**(ADM).

Disk load balancing

If your system is configured with more than one disk, it is important to spread the activity over all disks fairly equally. Use the **sar -d** command to get statistics about disk usage. The **sar**(ADM) manual page describes all the columns on the **sar -d** report; it is usually most interesting to look at the "Average" information printed at the bottom of the report. Numbers that are radically different between disk devices (especially in the "%busy", "avwait", and "avserv" columns) may indicate that file systems should be rearranged between your disks.

Note that **sar -d** reports on the disk activity occurring at the moment the data collector ran. If the data collector is running every 15 or 30 minutes and the disk load varies, individual numbers may be misleading. Nonetheless, the averages over a period of time can be very useful in evaluating whether your filesystems and swap device(s) are arranged to balance disk activity between all disks. See "sar -d" (page 515) for more information.

Specific steps to take in balancing data across all disks include:

- If you are running a large database application, put the database on a separate disk from the one that holds the *root* filesystem.
- Use the **swap**(ADM) command with the **-a** option to add swap devices to other disks. If your swapping activity is heavy, put additional swap devices on disks that do not contain other heavily-used files.

Filesystem troubleshooting

UNIX System V uses filesystems to store files on disk. Each filesystem corresponds to a disk division, which is identified as a file in the */dev* directory. Many system error messages identify a filesystem by the */dev* name rather than by the filesystem name that appears, for example, as output from the **df**(C) command. See Chapter 5 (page 95) in this guide for more information about administering filesystems; the sections here discuss common problems that may arise with filesystems.

Out of inodes on filesystem

An inode is an internal structure that the operating system uses to track 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: s5ialloc - Out of inodes on EAFS dev hd (nn/mm)
```

where ***nn/mm*** is the filesystem that has run out of free inodes. To fix this problem:

1. Remove unnecessary (old, temporary, *core*, or log files) files from the filesystem.
2. Use the **find**(C) command with the **-size** parameter to see if the filesystem contains a large number of small files. 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 the filesystem is created (using **mkfs**(ADM)). If the filesystem consistently runs out of free inodes, you can reconfigure the filesystem and increase the number of inodes. To do this:

1. Back up the filesystem using the **sysadmsh** selection:

   Backups ⇨ Create ⇨ Unscheduled

2. Verify the integrity of the backup using the **sysadmsh** selection:

   Backups ⇨ Integrity

3. Unmount the filesystem using the **sysadmsh** selection:

   Filesystems ⇨ Unmount

4. Run **mkfs** from the command line and specify more inodes for the filesystem.

   NOTE This procedure destroys the information on your hard disk. Do not use the **mkfs** command without a complete and verified backup.

   For example, to reconfigure the number of inodes on the */dev/u* filesystem to 6400, use the following **mkfs** command:

   mkfs /dev/u *fssize*:6400

   fssize must be replaced with the size of the filesystem in question, in 1K blocks. This value can be obtained as described in "Repairing a filesystem when fsck stops at size check" (page 571). For more information on the **mkfs** command, see the **mkfs**(ADM) manual page.

5. Mount the filesystem using the **sysadmsh** selection:

   Filesystems ⇨ Mount

6. Restore the filesystem from the backup with the following **sysadmsh** selection:

   Backups ⇨ Restore ⇨ Full

Out of space on filesystem

File data is stored in data blocks in the filesystem. Use the **df**(C) command to see the number of data blocks and inodes that are available. When a filesystem has little or no space left to work, the system displays the following message:

```
NOTICE: clalloc - Out of space on EAFS dev hd (nn/mm)
```

When the filesystem runs out of space, the system stops any attempts to write to the filesystem. The only way to restore system operation is to delete or reduce files from the named filesystem.

Use the following suggestions to restore space to the filesystem:

1. Use the **wall**(ADM) command to send a system-wide message asking users to remove unnecessary files.
2. Check for the number of blocks used by each file and directory in the named filesystem with the **du**(C) command. For example, the following command sequence displays the number of blocks in the directories under the */u* filesystem:

   cd /u; du

   Within a directory, the following command sorts the files by size and displays them with the largest first:

   ls -lsa | sort -nr | pg

   You can then send mail, asking the users who own the largest files to remove or compress them.
3. Use the **find**(C) command to locate exceptionally large or old directories and files, and send mail to the owners asking them to remove unnecessary files.
4. Use **find** to locate and remove temporary files, *core* files and any unused *a.out* files. Remove files from the */usr/preserve* and */lost+found* directories. In addition, you can modify the */etc/default/cleantmp* file and associated **crontab** entry that tells **cleantmp**(ADM) how often to clear key directories (*/tmp* and */usr/tmp* by default) of files.
5. Clear the contents of system log files, such as */usr/spool/lp/log*, */usr/adm/messages* and */usr/adm/sulog* (if enabled), and any log files for add-on programs. If disk space is tight, you can also delete data under the */usr/adm* directory, such as the **sar** data in the */usr/adm/sa* directory or the accounting data (if enabled) in the */usr/adm/acct* directory. This material can be first archived to tape or floppy if you think you may need to access it again.

   To clear a log file, use the following construction:

   Bourne or Korn shell:

   > *filename*

   C shell:

   cat /dev/null > *filename*
6. If you have security auditing enabled, check the disk usage in the */tcb/audittmp/audit** directories. Back up and remove old audit files using the **sysadmsh** selections:

   System ➪ Audit ➪ Files ➪ Backup

   System ➪ Audit ➪ Files ➪ Delete

   You should check and remove these files regularly.

7. Refer to "Reducing disk fragmentation" (this page) for the procedure to reduce disk fragmentation.
8. If the system is chronically short of free space, add additional hard disks and/or create and mount additional filesystems.

Each of these steps is described in more detail in the section on maintaining free space in filesystems in the chapter on managing filesystems in this guide.

Checking free space on filesystems

You should periodically check the amount of free space in your filesystems with the **sysadmsh** selection:

System ➪ Report ➪ Disk

This command prints the amount of space left in the filesystem in 512-byte blocks. If the space is low, see the previous section for hints on what to do (before you run out of space).

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 scattered widely across the hard disk. This results in increasingly poor disk I/O performance. Fragmentation may become a problem when the disk is (approximately) more than 75% full.

NOTE The following procedure cannot be used on the *root* filesystem.

To reduce disk fragmentation, first make a complete backup of all the files in the filesystem:

1. Invoke **sysadmsh** and select:

   Backups ➪ Create ➪ Unscheduled

2. Select the filesystem and media device (block size is selected automatically) and format floppy disks, if necessary.
3. Load a volume, tape or disk, into the selected drive, and press ⟨Return⟩. The system copies files to the drive.
4. Next, verify the integrity of your backup. Invoke **sysadmsh** and select:

   Backups ➪ Integrity

   Enter the media type and insert each volume of the backup in turn.
5. Use the **mkfs**(ADM) or **divvy**(ADM) command to recreate the filesystem.

6. Run **fsck**(ADM) to verify the integrity of the filesystem, then mount the filesystem:

   mount /dev/u /u

7. Restore the files from the backup. Invoke **sysadmsh** and select:

   Backups ⇨ Restore ⇨ Full

   Insert the first volume and enter the name of the filesystem that you want to restore and the media device. The actual **cpio** command line is displayed.

Because the files are completely rewritten on the disk, each file is written in one piece and fragmentation is reduced. It is a good idea to reduce disk fragmentation about once a year on a heavily-used system and less often on a lightly-used system. Verify that you have complete, accurate, and readable backups before you begin or you will lose files.

Restoring a corrupted root filesystem

If your *root* filesystem is so corrupted that, when you boot the system, **fsck**(ADM) cannot run, use the following procedure to restore your system:

1. Insert the boot floppy in the drive and reboot the machine.
2. At the Boot: prompt, press ⟨Return⟩. At the prompt, insert the *root* filesystem floppy. This boots the system and mounts the *root* filesystem from the floppies.
3. At the system prompt, enter:

   /etc/fsck /dev/hd0root

   You should see messages indicating that **fsck** is proceeding through five or six phases of system cleaning. If **fsck** exits within a few seconds or the system displays error messages that make no sense, you must restore the entire *root* filesystem from backups. Here are two examples of nonsense messages:

   ```
   UNKNOWN FILE SYSTEM VERSION 65535
   CLEANING NON SYSTEM 3 FILESYSTEM
   ```

   If **fsck** appears to be successful, shut down the system with **haltsys**(ADM) and boot from the hard disk by pressing ⟨Return⟩ at the Boot: prompt.

   If **fsck** is not successful, you must recreate the filesystem and restore it from backups. See "Restoring your root filesystem from backups" (page 569) for more information.

If you cannot boot from your hard disk at this point, you may have to reinstall the operating system from scratch. See the *Installation Guide* for more information.

Restoring your root filesystem from backups

To restore your *root* filesystem, you must have the following:

- Either a *root* filesystem backup created with **sysadmsh**(ADM) (either scheduled or unscheduled) or a full filesystem backup created with **cpio**(C).
- The Emergency Boot Floppy Set that you created immediately after installing your system.

> **NOTE** The Emergency Boot Floppy Set must have a kernel with the correct tape drivers installed. If the tape drivers are not present in the kernel on the Boot floppy, or if it is not configured with the correct parameters, the restore process will fail.

If you do not have an Emergency Boot Floppy Set, use the procedure described in "Restoring a root filesystem without an Emergency Boot Floppy Set" (page 570) to restore your *root* filesystem.

Restoring a root filesystem with an Emergency Boot Floppy Set

Use the following steps to restore your *root* filesystem:

1. If the system is not already down, bring it down. See "Stopping the system" (page 52).
2. At the prompt to reboot, insert your N1 (Boot) floppy disk in the drive and press ⟨Return⟩.
3. At the Boot: prompt, enter **defbootstr**.

   > **NOTE** If **defbootstr** results in an error, enter **restart**.

4. When prompted, insert the N2 (Filesystem) disk and press ⟨Return⟩.
5. Continue with the installation procedure to reinitialize your hard disk (**fdisk** and, depending on your disk, **dkinit**, **fdisk** and **badtrk**) and recreate the filesystems and swap space (**divvy**).

   For more information, see your *Installation Guide*.

   > **NOTE** If the hard disk or filesystem information is still valid, you do not need to recreate it. For example, if your hard disk still has a valid **badtrk** table, you do not need to run a **badtrk** scan.

   After you see the "Making Filesystems" message, you are prompted to insert the N1 disk at the Boot: prompt.
6. Insert the Boot floppy from your Emergency Boot Floppy Set and press ⟨Return⟩. **Do not** use the N1 floppy.

7. At the prompt, insert your Root floppy from your Emergency Boot Floppy Set and press ⟨Return⟩. **Do not** use the N2 floppy.
8. Enter:

   fsck /dev/hd0root

   This cleans the *root* filesystem on the hard drive.
9. Enter:

   mount /dev/hd0root /mnt

   This mounts the *root* filesystem on the hard disk.

   NOTE If you see the following message, you can safely ignore it:

   ```
   mount: warning /dev/hd0root   /mnt already in /etc/mnttab
   ```
10. Restore the filesystem with the following commands:

    cd /mnt
    cpio -ivmkBud -I/dev/*xxx*

    where ***xxx*** is either **rct0** (for a 1/4 inch cartridge tape) or **rctmini** (for a mini-cartridge tape).
11. Unmount and clean the newly restored filesystem with the following commands:

    cd /
    umount /dev/hd0root
    fsck /dev/hd0root
12. Shut down the system by entering **/etc/shutdown**.
13. At the prompt to reboot, remove the floppy from the floppy drive and press ⟨Return⟩ to boot from the hard disk.
14. Restore any secondary filesystems by entering **sysadmsh** and making the following selection:

    Backup ⇨ Restore

Restoring a root filesystem without an Emergency Boot Floppy Set

Use the following procedure to restore your *root* filesystem if you did not create an Emergency Boot Floppy Set at installation time. (You must install the Runtime System before you perform the restore.)

1. Perform steps 1 through 5 in "Restoring a root filesystem with an Emergency Boot Floppy Set" (page 569).
2. At the Boot: prompt, insert the N1 floppy and press ⟨Return⟩.
3. Continue with the installation procedure to install the complete Runtime System. See your *Installation Guide* for details.

4. At the prompt to reboot, remove the floppy from the floppy drive and press ⟨Return⟩ to boot from the hard disk.
5. At the "INIT: SINGLE USER MODE" message, enter the *root* password to enter system maintenance (single-user) mode.
6. Enter **mkdev tape** to install the tape drive. (See the chapter on adding tape drives in the *Hardware Configuration Guide* for more information.)
7. At the following prompt, enter "y ":

   ```
   The link kit is only partially installed. Do you wish to install it?
   ```

8. Insert the installation floppy disks when prompted.
9. Continue with the tape drive installation procedure and relink the kernel.
10. Enter **reboot** to reboot the system.
11. Verify that you can write and read from the tape drive.

    For example, insert a tape in the drive and enter:

    tar cv8 /etc/motd
    tar xv8 /etc/motd

12. Create the Emergency Boot Floppy Set using the following **sysadmsh**(ADM) selection:

    Filesystems ⇨ Floppy

    For more information, see the chapter on creating an Emergency Boot Floppy Set in your *Installation Guide.*
13. Shut down the system. See "Stopping the system" (page 52).
14. At the prompt to reboot, insert the Boot floppy from your Emergency Boot Floppy Set and press ⟨Return⟩.
15. At the prompt, insert your Root floppy from your Emergency Boot Floppy Set and press ⟨Return⟩.
16. Follow steps 8 through 14 in "Restoring a root filesystem with an Emergency Boot Floppy Set" (page 569) to restore your *root* filesystem (and any secondary filesystems) from backups.

Repairing a filesystem when fsck stops at size check

If the super block for a filesystem is so badly damaged after an abnormal shutdown or hardware failure that **fsck**(ADM) quits at the size check or reports an unusually large number of errors, you can use the **fsdb**(ADM) ("filesystem debugger") utility to patch the super block by hand.

The **fsck** utility reads the size of the filesystem from the super block. If the values currently stored in the super block for **FSIZE** (the total number of blocks in the filesystem) and **ISIZE** (the number of inodes allocated in the filesystem) are not the normal values for the filesystem, **fsck** displays the results of the size check and quits, as in the following example:

```
# fsck /dev/root
/dev/root
/dev/root File System: / Volume: root
Size check: FSIZE 0 ISIZE 0
#
```

When using **fsdb**, you have to change only one of the two values in the super block, **FSIZE** or **ISIZE**, back to their normal values for your filesystem.

NOTE You cannot run **fsdb** on the *root* filesystem if you are unable to boot up the system. If you have **fsdb** on the *root* floppy, you can use that version to repair the *root* filesystem.

Because **fsdb** is a powerful tool that allows you to directly change the super block (which contains important information about your filesystem), be very careful when using this tool. If you enter improper values with **fsdb**, you can permanently damage your filesystem and lose all filesystem data. The **fsdb**(ADM) manual page describes a number of other ways in which you can use **fsdb** to look at and manipulate the super block. The use of such advanced features is recommended only for the experienced system administrator.

NOTE Make sure that the filesystem is unmounted before using **fsdb** to repair it.

To repair the damage described above, follow this procedure:

1. Enter the following command:

   divvy

   NOTE When **divvy** is invoked without an argument, it uses the active partition on the first drive. For partition 2 on the first drive use **divvy /dev/hd02**. Refer to the **hd**(HW) manual page for a table of hard disk device nodes.

   Because the **divvy** filesystem table is not located in the super block, the information in the table is probably correct. From the **divvy** table, obtain the first and last block number of the filesystem that you are attempting to restore.

2. To obtain the correct value for **FSIZE**, apply the following formula:

   FSIZE = ***last_block*** – ***first_block*** + 1

3. To obtain the correct value for **ISIZE**, run **bc**(C), and enter the following command:

   ((FSIZE * .25) – ((FSIZE * .25) % 16))

   where **FSIZE** is the value computed in step 2. Round down any fractions and record this calculation of **ISIZE** for later use. Exit **bc** by entering **quit**.

4. To convert the number of available inodes into the actual address of the first block following the blocks allocated for inodes, you must perform an additional calculation. Refer to this number as **isize** to distinguish it from the **ISIZE** value displayed when **fsdb** starts.

   To do this, perform the following calculation:

   isize=(ISIZE / 16) + 2

   where **ISIZE** is the number that you calculated in step 3; you should round the value as previously described.

5. With your **FSIZE**, **ISIZE**, and **isize** values, you can now correct the super block values with **fsdb**(ADM). Enter the following:

   fsdb /dev/*filesystem*

   where ***filesystem*** is the name of the filesystem that you want to repair.

6. If the values that you calculated are close approximations of the real numbers, **fsdb** displays output similar to the following:

   ```
   FSIZE = 52985, ISIZE = 13232
   ```

   If the values match your calculation, you do not need to modify that parameter. If both values seem correct, something else is wrong and you must restore your data from backups.

   NOTE You can end the display of addresses at any time by pressing INTERRUPT or ⟨Del⟩.

7. Decide which parameter (or both), **isize** or **FSIZE**, must be corrected.

8. Enter the following commands (for XENIX, or UNIX system and AFS filesystems), substituting the values that you calculated earlier for **FSIZE** and **isize**. In the instructions that follow, your input is in bold and **fsdb** responses are in normal font, with real values in place of xxx and yyy. Remember to use the actual values for **FSIZE** and **isize**; do not enter the words "FSIZE" and "isize".

   If **isize** is correct, simply skip it by pressing ⟨Return⟩.

   NOTE Each time you press ⟨Return⟩, **fsdb** displays what the filesystem currently thinks the values of **FSIZE** and **isize** are.

To repair a XENIX filesystem, enter these commands:

```
1024⟨Return⟩
002000:  000000   (0)
=isize⟨Return⟩
002000:  000xxx
⟨Return⟩
002002:  000000   (0)
=FSIZE⟨Return⟩
002002:  000yyy
q
```

To repair an S51K, EAFS or AFS filesystem:

```
512⟨Return⟩
001000:  000000   (0)
=isize⟨Return⟩
001000:  000xxx
⟨Return⟩
001002:  000000   (0)
⟨Return⟩
001004:  000000   (0)
=FSIZE⟨Return⟩
001004:  000yyy
q
```

9. If the corrupted **FSIZE** value is so small that **fsdb** thinks that you cannot move that far into the block, the following error message is displayed when you input the beginning address:

   ```
   block out of range
   ```

   To disable error checking, enter a capital "O". You can then input the address without complaint.

At this point, you can run **fsck** on the filesystem and properly restore the system.

NOTE Currently, you cannot use **fsdb** to repair a DOS filesystem.

Filesystem mount failed

If the **mount**(ADM) command fails, check the filesystem with the **fsck**(ADM) command before running **mount** again:

/etc/fsck -y /dev/filesystem

where ***filesystem*** is the name of the filesystem that you want to check and repair.

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.

NOTE The contents of the files in the directory can be modified or removed, if the permissions allow write access.

Only *root* or the owner can set the sticky bit; only *root* or the owner of the directory can remove it.

To determine whether a directory has the sticky bit set, do a long listing of its parent directory (the sticky bit shows up as a "t" in the last field of the permissions listing). The following example shows a directory on which the sticky bit has been set:

```
drwxrwxrwt  5 sys      sys         2432 Jan 17 16:58 tmp
```

To remove the sticky bit setting from a directory, enter the following command:

chmod -t ***directory_name***

Users should now be able to remove files from the directory.

Login problems

This section covers what to do in situations in which:

- you cannot log in to multiscreens on the console after the system enters multiuser mode
- users cannot log into the system
- the system displays "Login incorrect" when a user tries to log in
- a user's account is locked
- a user forgot their password

Cannot log in after entering multiuser mode

If, after entering multiuser mode, you cannot log into the console even as *root* and the system displays the following error message:

```
Cannot obtain database information on this terminal
```

Refer to "Cannot obtain database information on this terminal" (page 598) for more information.

Users cannot log into the system

When the */etc/group* file is missing, users cannot log in and the system displays the following message on the user's terminal:

```
Can't rewrite terminal control entry for tty01.
Authentication error; See Account Administrator
```

The following error message is also displayed at boot time:

```
/bin/su:cannot setgid to auth, no auth entry
```

Refer to "Can't rewrite terminal control entry for tty" (page 597) for more information.

Login incorrect

If the system displays the "Login incorrect" error message when a user tries to log in, one of several problems may exist. Refer to "Login incorrect" (page 598) for more information.

useshell: file access control database inconsistency

If the system displays messages regarding "useshell" when creating or modifying a user, permissions on certain system files are incorrect. Refer to "useshell: File Control database inconsistency" (page 598) for more information.

Unlocking a locked user account

If a user account has been locked deliberately by the system administrator, the system locked the account because a user exceeded the number of unsuccessful logins attempts, or the user's password is expired, the system displays the following message when a user tries to log in to that account:

```
Account is disabled -- see Account Administrator
```

To unlock the account, invoke **sysadmsh** and select:

Accounts ➪ User ➪ Examine:Logins

Change the "Lock status" field to "Clear all locks". For more information, see "Locking or unlocking a user account" (page 68) in Chapter 4 (page 57) in this guide.

Replacing a forgotten user password

If a user forgets their password, the system administrator must change the password to a new one. The system does not provide a way to decipher an existing password.

To do this, follow these steps:

1. Invoke **sysadmsh** and make the following selection:

   Accounts ⇨ User ⇨ Examine:Password

2. Select "Change" from the Current password status selections.
3. Choose "Yes" from the Confirm Change form to select a new password. This invokes the password change procedure.
4. Select "1" (to select a password) or "2" (to have the system generate a password) and enter or choose a new password.
5. Press ⟨Return⟩ to return to the Modify User Account form.

For more information, see Chapter 4 (page 57) in this guide.

Mail troubleshooting

The MMDF system is configured for the local system by default. Therefore users should not experience problems with local mail. Generally, when **mail** does not work, there is either a problem with the network, or with MMDF configuration.

When mail problems occur, first verify that the network is working. To do this, test the appropriate network (for example, UUCP or TCP/IP). For information on testing the UUCP connection, see Chapter 13 (page 323) in this guide and "UUCP troubleshooting" (page 604). See your networking documentation for other testing information.

If UUCP is working and you still experience problems with **mail**, the problem is most likely with the MMDF configuration or the **deliver** daemon. The following sections discuss some common problems and solutions.

Failed mail error

If the system returns your mail immediately in a mail message with the words "Failed Mail" in the Subject line, there is a problem with the channel (**.chn*) or domain (**.dom*) files in the */usr/mmdf/table* directory. Use the following procedure to check this:

1. Log in as the **mmdf** user.
2. Change directories to */usr/mmdf/bin*.
3. Invoke **checkaddr** with the mail address that you used. For example, if **mail sylvia@seattle** failed, enter:

   ./checkaddr sylvia@seattle

4. If **checkaddr** displays "Unknown Host Domain," either the host or domain is not in the channel or domain files, or the */usr/mmdf/mmdftailor* file contains an error.

   If **checkaddr** displays "OK," then the host and domain are in the channel and domain files, but are incorrectly configured. Check the address with **checkaddr -w** to diagnose this problem.

5. Using the information in Chapter 14 (page 375) in this guide, verify that your domain, channel, and */usr/mmdf/mmdftailor* files are set up correctly.

Mail does not work, no returned mail

If mail does not reach its destination but MMDF does not return unsent mail, either **deliver** is not running properly, or the */usr/mmdf/mmdftailor* file contains an error. The **deliver** daemon is started automatically by a script in the */etc/rc2.d* directory at system startup; the process is owned by **mmdf**. Check to see that **deliver** is running using **ps -ef**. If **deliver** is not running, use the following procedure:

1. Log in as the **mmdf** user.
2. Start the **deliver** daemon manually with the following commands:

   cd /usr/mmdf/bin
   ./deliver [-c*channelname*] -b -T60

   where ***channelname*** is the channel that you want to check (for example, UUCP or SMTP); if you do not specifiy a ***channelname***, deliver checks all channels. This starts **deliver** for the specified channel sweeping the mail queue in the background every 60 seconds. For more information, see the **deliver**(ADM) manual page.

3. If running **deliver** manually does not work, check the configuration of the domain and channel files in */usr/mmdf/table* and the */usr/mmdf/mmdftailor* file.

Mail command hangs

If the **mail** command hangs when a user tries to read mail, check the following:

1. Verify that the user's mailbox exists in the */usr/spool/mail* directory. Unless your postmaster has configured the system otherwise, the */usr/spool/mail* directory should contain a file for each user who receives mail.
2. Make sure that the owner and group ID of the */usr/spool/mail* directory is **mmdf**.
3. Make sure that the mailbox is not too large (1000 messages is regarded as too large).

Modem problems

This section discusses common problems that you may have with your modem. Note that, while other serial ports are often used, the examples in the modem troubleshooting sections assume that the modem is attached directly to COM1. If you have problems, first verify that the phone jack is plugged in and that you have a dial tone on the phone line.

Modem — errors when dialing out

This section describes some situations and solutions to problems that may occur when dialing out on your modem.

The most useful tool for diagnosing dial-out problems is the **-x9** option to **cu**(C). This option causes **cu** to display diagnostic output when attempting to dial out. To get a debugging output, enter the command:

cu -x9 ***phone_number***

where ***phone_number*** is the phone number of the system you wish to dial.

No OK message (Hayes-compatible modems)

You get a connected message when you test the modem connection with the command:

cu -s1200 -l tty1a dir

but, when you enter **AT**, the system does not display the "OK" message. Use the following steps to solve the problem:

1. Verify that the modem switch and software settings are correct.
2. Check the modem cable:
   - If you are using a straight-through cable, try a null modem cable using at least pins 2, 3, 7, 8, and 20.
   - After issuing the **cu** command, watch the lights on the modem and press ⟨Return⟩ several times. The "receive" light should flash as you hit ⟨Return⟩. If it does not flash, check your cable to make sure that pin 2 is connected correctly (pin 2 is the data transmission line from the serial port to the modem).
3. If the "send" light flashes on the modem, the local echo may be turned off. Use the **ATE1** command to turn on the modem's echo capability.
4. Verify that the serial port on the computer is not defective:
   - Attach the modem to a different serial port, or attach a terminal or serial printer (with a proper cable) to the port to confirm that it is functioning.

- If the port is not functioning, check your hardware documentation for an appropriate repair facility.

5. If the previous steps do not fix the problem, your modem may be defective. If this is the case, check your hardware documentation for an appropriate repair facility.

Modem dials, but does not connect

If your modem dials okay, but the call never connects, check the following:

1. Verify that the phone number is correct and operational and that the phone line to which the modem is attached is not faulty. To do this, unplug the modem from the telephone line and plug in a regular telephone. Manually dial the number to make sure that the modem on the other end of the line is answering the call.
2. Listen carefully to your modem while it dials the call. Some business phone systems require a pause between certain numbers. Use a hyphen in the **cu** command to indicate a pause of two seconds. For example, if you enter: **9----458--1234**, the modem pauses 8 seconds after dialing the first number.

   The dialer translates the hyphen passed to the **cu** command into the appropriate code for your modem. For example, the dialer translates the hyphen into a comma before sending to a Hayes-compatible modem.

Connect failed: NO DEVICES AVAILABLE

When you try to dial out on the modem, the following message is displayed:

```
Connect failed: NO DEVICES AVAILABLE
```

Follow these steps to solve the problem:

1. Verify that the modem port has an entry in the */usr/lib/uucp/Devices* file. Here are example entries for a Hayes-compatible modem running at 2400 baud on */dev/tty1A*:

   ```
   ACU  tty1A  -  300-2400  /usr/lib/uucp/dialHA24
   Direct  tty1a  -  2400  direct
   ```

   Make sure that these lines in *Devices* do not begin with a number sign (#). There should be no spaces in front of the entries, and the "ACU" line must come before the "direct" line or **getty** will not hang up the modem correctly.
2. Verify that the modem port in *Devices* has the correct baud rate associated with it. If you specify the baud rate with the **-s** option to **cu**, verify that there is an entry in *Devices* that corresponds to that baud rate.

Modem answers, but terminal displays garbage

If the modem answers, but the terminal displays garbage characters, use the following steps:

1. Verify that the site that you are calling is set to the same data bit and parity values that you are using. By default, **cu** uses 8 data bits, and no parity. To change the values to 7 data bits and even parity, enter **cu -e**. For 7 data bits and odd parity, use **cu -o**, and use **cu -oe** for 7 data bits and no parity.
2. Verify that the remote computer is set to the same baud rate that you are using.

   If you are dialing into another UNIX system, you can force the remote site to switch to the next baud rate by sending a break signal. To send the break signal during the login sequence, enter:

   ~%b

3. Check for noise on your phone line. Noise problems become more acute when operating at 2400 baud or higher. Normally, when there is a problem with line noise, garbage characters appear on the screen in short bursts or continuously, as if a system on the other end of the line is trying to send valid data.

DEVICE LOCKED

Refer to "DEVICE LOCKED" (page 611) for more information.

Modem does not hang up

If your modem does not hang up at the end of a call, check the following:

1. Verify that you are using a modem control port that is configured in the */usr/lib/uucp/Devices* file and that the "ACU" entry for the modem port comes before the "Direct" entry for the direct line. If you are using a non-modem control port, change the port to the corresponding modem control port. For example, the modem control port associated with *tty1a* is *tty1A*.

   NOTE Non-modem control ports should only be used with terminals, and when configuring the modem.

2. If the CD (Carrier Detect) light on the modem does not go off when the call is disconnected, check the modem switches to verify that the modem is set to detect the incoming carrier. If your modem is a Hayes 2400 or compatible, use the **AT&C1** command. This forces the carrier detect line to follow the presence of a carrier on the phone line.

3. Check the modem switches to verify that the modem is set to detect DTR (Data Terminal Ready). The modem should hang up when DTR goes from high to low. If the modem is a Hayes 2400 or compatible, use the **AT&D2** command.
4. Some modems have a switch that can be set to ignore DTR; make sure that this switch is off.

Double echo

If you get a double echo when you dial out on your modem, check the setting for local echo. If local echo is enabled, disable it.

Modem — problems dialing in

This section provides solutions for common problems that may occur when dialing in on your modem.

Modem does not answer the phone

If the modem does not answer the phone, check the following:

1. Verify that the modem control port is enabled. To enable the modem port, enter the following commands:

   disable /dev/tty1a
   enable /dev/tty1A

2. Verify that the modem is configured to auto-answer. Check your modem switches. If the modem is a Hayes 2400 modem, enter:

   cu -l tty1a dir

   Then, use the **ATS0=1** command to tell the modem to answer the phone on the first ring. (Remember to enter **AT&W** to save modem settings.)
3. Verify that the DTR (Data Terminal Ready) line is connected from the computer to the modem. Make sure that Pin 20 is connected. Pins 2, 3, 7, 8, and 20 are required for modem communication.
4. Make certain the ACU entry for this modem in the *Devices* file precedes any "Direct" entries for the port or the non-modem control counterpart.

Modem answers, but hangs up

If the modem answers, but hangs up immediately upon connection, check the following:

1. If the modem is set to auto-answer and to detect DTR, check to see that the DTR line is asserted.

2. Verify that the modem control port is enabled:

   disable /dev/tty1a
   enable /dev/tty1A

3. Verify that the cable is correct. If you are using a straight-through cable with at least pins 2, 3, 7, 8 and 20 connected, verify that pin 20 (DTR) is properly connected.
4. Check to see if the modems have data compression or error correction modes set. After a connection is established, modems that support special error correction or compression protocols attempt to "negotiate" which of these protocols to use. If the other modem is programmed not to use any of these modes (or does not support them), it may misinterpret this negotiation as actual user input. In general, modes such as MNP4 or v.42bis should only be enabled when connecting with a modem with the same modes enabled. Check the documentation for your modem.

Garbage or loose cable

The console displays a message like the following when a call comes into the modem:

```
Garbage or loose cable on /dev/tty1A, port shut down
```

Check the following:

1. Verify that your modem is not set to echo back data or send command responses. If the modem is not set up this way, it may be sending a "RING" signal to indicate that the phone you are calling is ringing. Because the CD signal is not active, the serial driver interprets this as random data on the serial line. The proper Hayes 2400 modem command is **ATE0Q1**.
2. If you have an internal modem and the above options do not eliminate the error message, your modem may be incompatible. Contact the manufacturer to see if a fix is available. If no fix is available, you may need to replace your modem with a standard Hayes-compatible external modem.

Modem answers, but no login prompt

If the modem answers, but does not display a login prompt, check the following:

1. Verify that the CD line is being asserted by the modem after the modem has answered the phone. Check the switches on your modem or, if your modem is a Hayes 2400 or compatible, use the **AT&C1** command. (Remember to enter **AT&W** to save modem settings.)

2. Make sure that the port is enabled. Enable the port by entering the following command sequence:

   disable /dev/tty1A
   enable /dev/tty1A

3. Verify that the modem is using the correct */etc/gettydefs* entry and is selecting the proper baud rate. The modem port device line in the */etc/inittab* file should look like the following:

   ```
   t1A:2:respawn:/etc/getty -t60 tty1A 3
   ```

   The last field on the line is the pointer to the entry in the */etc/gettydefs* file. In this case the */etc/gettydefs* entry is "3", which is the 2400-1200-300 baud entry. Verify that this entry in */etc/gettydefs* is correct.

Screen displays a series of login prompts

If the screen scrolls uncontrollably when you log in, usually displaying a series of login prompts, verify that only the modem device is enabled. If the non-modem device is enabled, disable it:

disable /dev/tty1a

System displays meaningless characters

If the system displays the login prompt, but no password prompt, or meaningless characters are displayed after the login prompt, verify that the line settings are correct:

1. Determine the serial line settings on the system that you are calling. The standard settings that **cu** uses are 8 data bits, one stop bit, and no parity.
   - If the remote system uses even parity, use **cu -e**.
   - If the remote system uses odd parity, use **cu -o**.
   - If the remote system uses 7-bit no parity, use **cu -oe**.
2. If you are dialing into a UNIX system, check the */etc/inittab* file on the remote system to verify that the "pointer" into the */etc/gettydefs* file is correct. The serial line characteristics may not match between the stty settings defined in the third field of the selected *gettydefs* entry. Change the setup for the port to 8 data bits, one stop bit, and no parity.

   The entry should similar to the following:

   ```
   3 # B2400  HUPCL OPOST CR1 ECHOE NL1 #
          B2400 CS8 SANE HUPCL TAB3 ECHOE IXANY #\r\n@!login: # 1
   ```

Mouse (serial) troubleshooting

The error messages in this section are associated with mouse problems. If you only recently installed the mouse, make certain you checked all the points covered in the chapter on adding mice and other graphic input devices in the *Hardware Configuration Guide.*

usemouse: no mouse available(5)

Try executing the **tty**(C) command on the multiscreen that is failing. If the **tty** command is executed on the first multiscreen, it will normally return "/dev/tty01". If **tty** returns "/dev/syscon" on the failing multiscreen then the system was shutdown on that multiscreen using the **init s** or **init S** command. Shutting down the system with **init s** or **init S** will cause the system to enter single-user mode. As documented in the **init**(M) manual page, this will also cause the system console device (*/dev/syscon*) to remain linked to the tty from which the last **init S** was invoked.

To correct this problem, and allow the mouse to work, take the following steps:

1. Log in as *root* and run **mkdev mouse**.
2. Select option 4, "Associate a terminal with an existing mouse."
3. Enter **syscon** as the terminal device.
4. Enter **exit** to exit from **mkdev mouse**.

This adds */dev/syscon* to the list of terminals with which the mouse can be used.

usemouse: can't open slave (errno 13)

This message means that **usemouse** cannot open the slave side of the pseudo-tty that is necessary to communicate with the system.

Check the permissions on the "master" pseudo-ttys (devices beginning with */dev/ptyp*) and "slave" pseudo-ttys (devices beginning with */dev/ttyp*). They should be 666 mode or "rw-rw-rw-". If they are not correct, change them using the **chmod**(C) command. As a general rule, if a problem is related to permissions, running as a normal user gives the problem, but running as *root* does not. In addition, check the entries in the file */usr/lib/event/ttys* which indicate that a mouse is a legitimate "event source" for the tty you are on. For example:

```
tty01 msbusmouse
```

This entry indicates a Microsoft busmouse is configured as usable on the first console multiscreen.

No more processes error message

The "No more processes:" error message is displayed when you attempt to execute a process that would exceed some system resource. This error message can indicate one of several conditions, including:

- The memory required to run all the user processes is more than the total available main memory and swap space. At least 10% of available memory should be free at any given time. To check this, use the **crash**(ADM) command. Within **crash**, get the values of the **availsmem**, **nswap**, and **maxmem** kernel variables by using the **od -d** ***variable*** command. The following equation determines the percentage of usable memory that is still available:

  availsmem / ((***nswap*** / 8) + ***maxmem***)

  This quotient should be greater than 0.10 at all times. Larger numbers are better. See "Understanding your performance goals" (page 490) for a discussion of ways to resolve this problem.
- You have run out of entries in the process table whose size is determined by the **NPROC** tunable parameter. The **sar -v** command gives statistics for the process table.
- You are running more processes than the system allows per user. Try scheduling some jobs to run at a different time. If this is not possible, you may need to adjust the value of the **MAXUP** tunable parameter. See **MAXUP** (page 536) for more information.

Runaway processes

A runaway process is a process that enters an infinite loop and spawns new processes. This can cause an overflow in the **proc** table that causes other processes to fail with the "No more processes:" error message. If the **sar -v** report shows that the kernel proc table has overflowed, use the **ps**(C) command to check for a runaway process. To display all the processes on the system, use the **ps**(C) command, such as the following:

ps -ef | grep -v getty | more

This displays all executing processes except for **getty** lines that represent idle terminals. Look for processes that have consumed large amounts of time, or several similar processes that may have been spawned by a runaway process that is in an infinite loop.

See "Runaway processes, stopping" (page 594) for information about how to stop a runaway process.

Print system troubleshooting

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 request, or does not start any new print jobs, determine if the **/usr/lib/lpsched** daemon is running. To do this, enter the following command:

lpstat -r

(The **lpsched** daemon is started automatically by a script in the */etc/rc2.d* directory at system startup.)

If the scheduler is down, the system displays the following message:

```
scheduler is not running
```

If this is the case, use these steps to start the **lpsched** daemon:

1. Log in as the super user *root* or a user with **lp** authorization.
2. Invoke **sysadmsh** and select Printers ⇨ Schedule ⇨ Begin. The system displays the following message:

   ```
   Print services started
   ```

It may take a minute or two for the printer configurations, forms, and filters to be re-established before any saved print requests start printing. Any print requests that did not finish printing when the scheduler stopped are printed in their entirety when the print service restarts. The printer configurations, forms, and filters in effect when the print service stops are restored when it restarts.

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. Check the printer documentation for a self-test.
4. Check the printer cable and make sure that it is attached properly to the port and the printer. Refer to the owner's manual for your printer 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 for install-

ing a printer in the *Hardware Configuration Guide.*

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 590).

6. Make sure that the printer is enabled. To do this, invoke **sysadmsh** and select:

   Printers ➪ Schedule ➪ Accept

   Printers ➪ Schedule ➪ Enable

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 590) 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. Refer to the *Hardware Configuration Guide* 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 normally include **opost** and **onlcr**.

If the output from the redirected command does not print, follow the instructions in "Cannot redirect output to printer" (page 589).

If the output prints, try submitting a sample file (like */etc/motd*) for printing by invoking **sysadmsh**, and selecting:

Dirs/Files ➪ Print

If the hardware connections are good and the printer is properly configured and enabled, but is still idle and print requests are queued, check the following:

1. Issue the **lpstat -r** command to verify that the **lpsched** process is running.
2. Restart the **lpsched** daemon if it is not present:

   /usr/lib/lpshut
   /usr/lib/lpsched

3. Check to see that print requests 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.
5. Check to see if a dialout printer was busy or did not answer, or all dialout ports are busy. The print service waits five 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, check the following:

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 *lp0*:

   ls -l /dev/lp0

   The output should look similar to this:

   ```
   crw-------   2 bin      bin        6,  0 Jun 13 1992 /dev/lp0
   ```

   A text file would be missing the "c" at the beginning of the line.
2. Test the cable connection using a cable from a working 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, check the following:
   - Verify that the port is configured correctly; see "Port does not respond" (page 590) for more information.
   - 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 by using **hwconfig** (for example, **unit=0** corresponds to *lp0*.

- If you still cannot print using a different device name, your printer may be defective; check the hardware documentation that came with your printer.

4. If you cannot print from DOS, check the printer hardware configuration. See the documentation that came 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, check the following:

1. Verify that the parallel or serial card is properly seated:
   - Turn the power off and open the machine.
   - Remove and reseat the card in the bus.
2. Run **mkdev serial** or **mkdev parallel** to see if the port is listed in your kernel configuration.
3. Verify that the card itself is correctly configured; check the documentation that came with the card. If possible, try setting the card for a different configuration.
4. The card may be defective; replace it.

Printer output is illegible

If the printer prints illegible output, check the following:

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.) Set the baud rate to 9600 baud for optimum performance.

   To set a different baud rate for the print service to use, use the following command:

   Printers ⇨ Configure ⇨ Parameters

   In the "Default initial settings" section, enter the baud rate number in the "stty" field. Then submit a sample file for printing.
2. Determine the parity setting for the printer and check to see that it 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, use the following **sysadmsh** selection:

Printers ⇨ Configure ⇨ Parameters

In the "Default initial settings" section of the form, add one of the following to the "stty" field:

oddp Sets odd parity generation.
evenp Sets even parity generation.
-parity Sets no parity (default).

Select the option that matches what your printer requires.

3. Check to see that the tabs are set correctly. Refer to "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 type is correct. If the wrong printer type was selected when you set up the printer with the print service, the wrong control characters can be sent to the printer. This situation can cause output to disappear or be illegible. A simpler problem to solve is when it sets the wrong character set or font.

If you do not know the printer type, examine the */usr/lib/terminfo* directory for a list of names that are available. Look for names that match one word or number identifying your printer. For example, the name 495 identifies the AT&T 495 Printer.

When you have the name of a printer type that you think is correct, change the printer type setting. To do this:

1. Invoke **sysadmsh** and select:

   Printers ⇨ Configure ⇨ Parameters

2. In the printer name (terminfo database) field, enter the printer type name.

Printer output spacing is wrong

If the printer output is legible, but the spacing is wrong, invoke **sysadmsh** and select:

Printers ⇨ Configure ⇨ Parameters

For each of the situations, adjust the following settings in the "stty" field of the "Default initial settings" section:

- If the printer output is double-spaced, enter either the **-onlcr** or **-tabs** option.
- If there is no left margin and the text runs together, enter the **-tabs** option.
- If the printer output zig-zags down the page, enter the **onlcr** option. (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 using the the instructions for installing a printer in the *Hardware Configuration Guide.* In particular, printing can be slow if the port is configured for the right I/O address but the wrong interrupt vector. To check this, run **mkdev parallel** and note the interrupt used by the slow port. Use the remove option, and then add it once again with the same I/O address, but switch the interrupt (5 instead of 7, or vice versa). Relink, reboot, and see if the port is still slow.

If printing is still slow after verifying that the parallel ports are configured correctly, the problem may be that your parallel port is not capable of generating interrupts. In this case, you can try setting up polling for your printer port. If your 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, there is a kernel parameter that changes this behavior. Each solution is described in the sections that follow.

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, create what is known as a *special device node.*

Use the following procedure:

1. Use the **shutdown su** command to bring the system into maintenance mode.
2. Note which parallel printer ports are configured. (Check */usr/adm/messages* or run the **/etc/hwconfig -h** command.) For example:

   ```
   parallel 0x378-0x37A   07    -    unit=0
   ```

   where "unit=0" refers to *lp0.*
3. Configure a special device file for the printer by creating 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 the following commands:

   cd /etc/conf/cf.d
   touch /etc/.new_unix
   ../bin/idmkenv

5. You see the following:

```
The kernel environment includes device node files and /etc/inittab.
The new kernel may require changes to /etc/inittab or device nodes.
Do you want the kernel environment rebuilt? (y/n)
```

   Enter " y " and press ⟨Return⟩.

6. The following is displayed:

```
The kernel has been successfully linked and installed.
         To activate it, reboot your system.

Setting up kernel environment
```

7. Enter the **haltsys** command to reboot your system. Reboot the system as instructed and enter multiuser mode. The new polling device is now in place.
8. If you are using the print spooler, you must now inform the spooler of the new parallel poll device. Invoke **sysadmsh** and select:

   Printers ⇨ Configure ⇨ Modify

9. In the "Device name" field, do not select a standard parallel device name. Instead, use either */dev/lp0p*, */dev/lp1p*, or */dev/lp2p*.

Changing the MODE_SELECT parameter

If you suspect that your printer is deselecting and reselecting itself after each line, you can change the value of the **MODE_SELECT** kernel parameter. (This works with both polled and standard **lp** devices.) **MODE_SELECT** has a default value of 1 which enables mode-select checking. Setting this value to 0 disables mode-select checking, and hences remove the printing delay incurred by these checks.

To change the value of **MODE_SELECT**, use the **configure**(ADM) utility. The **MODE_SELECT** parameter is found in the "Hardware dependent parameters" category of the **configure** menu. See "configure(ADM) — reallocate kernel resources" (page 504) for complete instructions.

NOTE Slow printing has been observed on some models of Tandy® printers, but is not found exclusively on Tandy printers.

Printer reports UUCP errors

If UUCP is configured, the print service uses the UUCP software to handle dialout 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 "UUCP error messages" (page 369) in Chapter 13 (page 323) in this guide and "UUCP troubleshooting" (page 604) for more information.

Runaway processes, stopping

A *runaway process* can cause an error that locks up the keyboard, preventing anything that the user types from reaching the system. Because of this, a runaway process cannot be stopped from the terminal at which it was invoked.

To continue, you must identify and stop the runaway process:

1. Log in as *root* on a terminal (or console screen) that is not locked.
2. Enter the **ps -ef** command and press ⟨Return⟩. The system displays full information on all the current processes and their process identification numbers (PIDs). Find the PID of the runaway program. You may see on program that is consuming more CPU time than is reasonable, or may find a large number of processes with a common parent who are running the same program.
3. Enter the **kill** *PID* command. The program should stop in a few seconds. If the process does not stop, enter the **kill -9** *PID* command.

The last command may leave temporary files that are usually removed when a program terminates normally, or a non-echoing terminal. Refer to "Restoring a non-echoing terminal" (page 604) to restore the terminal to normal operation. If the runaway process does not stop when you enter **kill** with the **-9** option, the process is considered *unkillable*. If the unkillable process is a user's shell, you must stop the process before that user can continue working. To stop an unkillable process, you must reboot the system. Use the following procedure:

1. Log in as *root* and send a message using **wall**(ADM) to the other system users notifying them of the impending shutdown.
2. When all the users have logged out, shut down the system by entering **init 6**. The root prompt (#) is returned, but the shutdown process begins soon afterward.

3. Reboot the system by pressing ⟨Return⟩ at the Boot: prompt.

The unkillable process no longer appear when you run **ps**.

Security system troubleshooting

This section lists problems and error messages that you may encounter. Each problem is discussed in context, including the reason for the situation, the solution to the problem, and ways to prevent the situation from recurring. See Chapter 4 (page 57) in this guide for more information about administering the security software.

You must log in as *root* to fix the problems discussed here. In some cases, you must use the "override" terminal to fix problems with the security system. This feature is discussed first.

Using the override terminal

If your system is running with High security defaults, an override terminal entry exists for *root*. This terminal can be used if the security databases become corrupted so that all logins are disallowed. The */etc/default/login* file contains an entry that identifies which terminal (tty) is to be used for 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.

```
OVERRIDE=tty01
```

You can change this default to be another login device.

When the databases are compromised and *root* logs in on the override terminal, the following message is generated:

```
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, the following message is generated:

```
Account is disabled but console login is allowed.
```

The device used as the override terminal should be physically secure; remember that normal locks do not apply to the super user account on this tty.

If you are locked out and you removed the default override tty, you must reset the system or power cycle it, and come up in single-user (maintenance) mode to enter the system. Because shutting off the system can lead to file-system damage, it is critical that you have an override terminal.

Account is disabled -- see Account Administrator

If you see this message, it means the account is locked for one of three reasons:

- You locked the account through the **sysadmsh** selection Accounts ⇨ User ⇨ Examine:Logins. If you want to re-establish the account, use the same selection to unlock the account.
- 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 or use the Accounts ⇨ User ⇨ Examine:Logins selection to clear the lock (in which case the user is forced to set the password at login 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 lock-out. It may be that the user is a poor typist: or another person is trying to lock the account and knows this is a way to do it, or 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 super user 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 super user account does not prevent the super user from logging into the console. This presents a means for entering the system even when all other accounts or terminals are locked. Before continuing, use the audit trail to investigate the reasons for the lock. A lock-out caused by unsuccessful login attempts on the system console is cleared automatically, but lock-outs due to other reasons remain in effect. The console, in effect, is never locked out for the super user.

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 **sysadmsh** System ➪ Audit ➪ Disable selection to terminate auditing if it has not already been done. Analyze the source of the problem and solve it before re-enabling auditing.

Authentication database contains an inconsistency

This message is displayed while running one of the programs associated with the TCB or a protected subsystem. The Authentication database integrity is in question. The Authentication database is a composite of the Protected Password database, the Terminal Control database, the File Control database, the Command Control database, the Protected Subsystem database, and the System Defaults file, and the message applies to all of these. Either a data entry is not present when expected, or the items within an entry are not correct. This message is 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 Database Events were enabled for the user that generated the message. The Accounts ➪ Check ➪ Databases selection 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 tty entry in the */etc/auth/system/ttys* file is corrupted. If a multiscreen entry in the */etc/auth/system/ttys* file is corrupted, the multiscreens listed below the corrupted entry are inaccessible. For example, if the entry for *tty03* is corrupted, when you press ⟨Alt⟩⟨F4⟩ or ⟨Alt⟩⟨F5⟩, the system displays a blank screen.

To solve this problem:

1. Log in on the override tty.
2. If the */etc/auth/system/ttys* file does not contain an entry for *tty01*, add the following line below the console entry:

   ```
   tty01:t_devname=tty01:chkent:
   ```

   You can now enter multiuser mode and access the multiscreens of the console.
3. If the */etc/auth/system/ttys* file appears intact, you should check and see if the */etc/group* file was corrupted or removed. If so, restore the file from backups.

Cannot obtain database information on this terminal

This message indicates a problem with the */etc/auth/system/ttys* file. Running the **tcbck**(ADM) utility should fix any problems with the *ttys* file. Under normal conditions, **tcbck** is run automatically whenever the system boots.

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. Log in on the override tty and restore the file from backups.

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 with **sysadmsh** or a number of incorrect login tries (to one or more accounts) passed the threshold for that terminal. In both cases, determine what happened and then use the **sysadmsh** to reset the lock.

useshell: File Control database inconsistency

This message occurs while attempting to add users with **sysadmsh** Accounts ➪ User ➪ Create ➪ . This means there is a problem with the permissions or ownership of a database file located in */usr/lib/mkuser*. You should run the **fixmog**(ADM) utility as follows:

fixmog -i

The **-i** option checks files interactively, asking you to to confirm each change before it is done. There may be additional problems reported by **fixmog** that should be examined via an **integrity** report; see "integrity(ADM)" (page 509) for more information.

useshell: Script path /usr/lib/mkuser/mkuser.init may be compromised ...

This means there is a problem with the permissions or ownership of a database file located in */usr/lib/mkuser*. You should run the **fixmog**(ADM) utility as described for the previous item.

You do not have authorization to run ...

The command is part of a protected subsystem. For that subsystem, the authentication administrator has not provided you with the kernel authorization needed to run this command and/or related commands. The authentication administrator uses the Accounts ➪ User ➪ Examine:Privileges selection to grant or deny such authorizations.

Tape drive problems

This section covers some common problems that you may have with your cartridge, floppy, or SCSI tape drive on your system. Note that these problems generally occur immediately after using **mkdev tape** to install your tape drive.

For more information about installing and configuring tape drives, see the *Hardware Configuration Guide.*

Bad octal digit

After you run **mkdev tape** to install a cartridge tape drive during the kernel relink process, the system displays the following message:

```
/etc/conf/pack.d/ct/space.c line 46 bad octal digit
```

To fix this problem, verify that the base address for the tape controller is entered with a leading "0x" rather than a trailing "H". Run **mkdev tape** again to modify the cartridge tape parameters.

Tape not recognized at bootup

If your system does not display a message like the following at boot time, the system does not recognize the tape controller card:

```
%tape          0x338-0x33C    05          1      type=wangtek
```

Instead of this hardware recognition message, the system displays a message like the following:

```
ct: ERROR: Tape controller (type=wangtek) not found
```

Cartridge tape

For a cartridge tape drive, check the following:

1. Verify that the tape controller card is physically configured to the base address that you gave when you ran **mkdev tape**. If necessary, run **mkdev tape** again and change the parameters. For information about configuring your cartridge tape drive, consult the *Hardware Configuration Guide.*

2. Verify that the tape controller is seated properly on the motherboard and, if necessary, insert the controller in a different slot.
3. Some tape drives (particularly external drives) require that the drive be attached to the tape controller and powered on at boot time.
4. Verify that the cartridge tape drive is supported. See the SCO list of compatible hardware for a list of the compatible tape drives.
5. If your system still does not recognize the tape controller card at boot time, see the documentation that came with your hardware.

Irwin and QIC-40/80

For an Irwin (mini cartridge) or QIC-40/80 tape drive, verify that the settings you supplied when you ran **mkdev tape** actually match the jumpers set on the back of the floppy tape drive.

SCSI tape

For a SCSI tape drive, run **mkdev tape** to check the following:

1. Verify that the ID number for the controller of the device is correct. The ID number is determined by the jumper settings on the controller. The valid range is 0-7. SCSI tape drives are often configured for ID.
2. Make sure that the host adapter number is correct. The first SCSI host adapter of a given type is 0; the second is 1.
3. Check that the LUN (Logical Unit Number) is correct. In most cases, the controller is embedded in the same physical unit as the device and supports one device with LUN 0. If the controller is not embedded, it supports up to eight devices. If this is the case, the LUN is determined by the jumper settings on each device. The valid range is 0-7.
4. Verify that the host adapter itself is recognized at boot time.
5. Verify that the tape drive is supported. Refer to the SCO list of compatible hardware for a list of supported tape drives.

Tape commands hang

If the tape drive hangs when you test it with the following command:

tape reset

Check the following:

1. Verify that the cartridge tape controller card is physically configured for the DMA value that you gave when you ran **mkdev tape**. To fix this, either physically reconfigure the tape controller DMA setting to agree with the address that you gave with **mkdev tape**, or run **mkdev tape** again and specify the DMA value on the tape controller. Do the same for the interrupt vector.

2. Verify that the cable between the tape controller and the tape drive is properly attached.
3. If the **tape reset** command still hangs, see the documentation that came with your hardware.

Cannot open /dev/rct0 error message

If a **tar**(C) command or other commands which access the tape drive fail to write the contents of a directory to a tape and the system displays the following message:

```
Cannot open /dev/rct0
```

Check the following:

1. Use **hwconfig** to verify that the interrupt vector for a cartridge tape drive does not conflict with any other device on your system. Refer to "Tape not recognized at bootup" (page 599) for information on the available interrupt vectors. To change the interrupt vector, run **mkdev tape** and indicate a new interrupt vector value.
2. Verify that the actual physical interrupt (IRQ) setting on a cartridge tape controller card agrees with the interrupt vector that you specified with **mkdev tape**.
3. Make sure that the proper device for the tape drive is located in the */dev* directory. Enter:

   l /dev/rct0

   The output should look like the following:

   ```
   crw-rw-rw   1 root  root  10, 0 Feb 14 12:00 rct0
   ```

   (The major device number (10) may differ.) If the listing of */dev* does not contain a line similar to this, run **mkdev tape** again to create the device.
4. Verify that all cables are connected correctly.
5. If the **tar** command still does not work, the tape hardware may be broken. See the documentation that came with your hardware for more information.

Terminal troubleshooting

This section explains how to solve the following problems with terminals:

- terminal is completely non-functional
- terminal is hung
- terminal display is scrambled

- terminal is locked
- terminal does not echo

Restoring a non-functional terminal

A completely non-functional terminal displays no login prompt and does not respond to keyboard input. This situation is usually caused by hardware failure or configuration problems. Follow these steps to check a non-functioning terminal:

1. Check the brightness control on the terminal.
2. Check the power and communication connections at the terminal and computer.
3. If applicable, enter set-up mode on the terminal and verify the terminal configuration settings. The settings should include 9600 baud, 8 data bits, 1 stop bit, and no parity.
4. Enable the port to which the terminal is connected. For example, to enable *tty007*, use the following command:

   enable tty007

5. Verify that there is a **getty** process associated with the terminal port. For example, enter:

   ps -t tty007

6. Test the hardware communications by disabling the port and redirecting output to the non-functional terminal: for example, to test *tty007*, use the following commands:

   disable /dev/tty007
   echo hello > /dev /tty007

If these steps do not restore the non-functional terminal, check the documentation that came with the terminal hardware for troubleshooting suggestions.

Fixing a hung terminal

A terminal is considered "hung" if the previous work session is still visible on the display, but it does not respond to keyboard input. Use the following steps to fix a hung terminal:

1. Wait a minimum of 60 seconds before trying to resurrect the terminal. (If the system is busy, the terminal may not respond immediately to keystrokes because the system response time has increased.)
2. Press ⟨Ctrl⟩**q** to re-enable transmission in case the ⟨Ctrl⟩**s** (transmit off) signal was inadvertently pressed.

3. Check to see that all power cords, keyboard cords, and communications cables are connected.
4. Reset the terminal hardware by recycling power to the terminal and then reinitialize it by running **tset**(C) with no arguments.
5. Verify the terminal set-up mode configuration settings (if available) as described in step 3 of the previous section.
6. Test the hardware communications by redirecting output from an operating terminal to the locked one as described in the step 6 of previous section.
7. Check the processes that are running on the locked terminal port with the following command:

   ps -t tty*n*

   Stop the process that the user was running when the terminal hung using **kill**(C). Refer to "Runaway processes, stopping" (page 594) for information on how to do this. If the program does not die, you must reboot the system to stop the process.
8. Determine whether the current line characteristic parameters are correct. For example, use the following command to display these values for *tty007*:

   stty -a < /dev/tty007

   You can also compare the **stty** settings with those of a working terminal.
9. Reset the serial line characteristics with the following command:

   ⟨Ctrl⟩**j stty sane** ⟨Ctrl⟩**j**

   If you cannot enter the command on the terminal, you can redirect the **stty** command from another terminal as follows:

   stty sane < /dev/tty007

If the **ps -t** command shows only a **getty** program, the terminal should display a login prompt. If it does not, check the terminal hardware again.

Fixing a scrambled terminal display

A scrambled terminal responds to keyboard input but the display is incorrect. Follow these steps to fix a scrambled terminal:

1. Check the terminal type (**TERM**) for the user with the **env** command. If the terminal type is incorrect, reset it. For example, to set the terminal type to "wyse60", enter:

   TERM=wy60

   After resetting the terminal type, reinitialize the terminal by entering **tset** with no arguments.

2. Reset serial line characteristics with the following command:

   ⟨Ctrl⟩**j stty sane** ⟨Ctrl⟩**j**

Unlocking a locked terminal

If a terminal has been locked by the system administrator to prevent logins on that terminal, or if the system locked the terminal because a user exceeded the number of unsuccessful logins attempts, the following message is displayed on that terminal:

```
Terminal is disabled -- see Account Administrator
```

To unlock the terminal, invoke **sysadmsh** and select:

Accounts ⇨ Terminals ⇨ Unlock

Restoring a non-echoing terminal

A non-echoing terminal is a terminal that responds to keyboard input but does not display the characters entered at the keyboard. (This is different from a *locked* or non-functional terminal that does not respond to input at all; see "Unlocking a locked terminal" (this page) or "Restoring a non-functional terminal" (page 602) for a solution to these problems.)

Sometimes, when a program stops prematurely as a result of an error, or when the user presses the ⟨Break⟩ key, the terminal stops echoing. To restore the terminal to normal operation, enter the following:

⟨Ctrl⟩**j stty sane** ⟨Ctrl⟩**j**

Enter this command accurately because the terminal does not display what you enter at the keyboard.

The terminal should now display keyboard input. If it does not, follow the steps outlined in "Restoring a non-functional terminal" (page 602).

UUCP troubleshooting

This sections discusses common problem situations that you may experience with UUCP and covers the following topics:

- how to check for a faulty ACU/modem
- error messages that **cu** displays
- common errors when running **uucp**
- how to check the status of a **uucp** request
- using **uutry** to debug UUCP communications

- common messages that appear in the UUCP log and status files
- how to check permissions settings on UUCP files
- checking that your sitename is unique
- what to check if UUCP is very slow
- what to do when **uucp** works, but **uux** does not
- a description of the UUCP troubleshooting utilities, **uucheck**, **uulog**, **uuname**, **uustat**, and **uutry**

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. If you are using a direct line, make sure that the computers are connected correctly. Use the instructions in Chapter 12 (page 305) and the section "Connecting two local systems using a direct wire" (page 360) in Chapter 13 (page 323) in this guide.

NOTE If you are configuring an outbound UUCP connection over a modem, **cu** must function in order for UUCP to work.

Check for faulty ACU/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*

Refer to "Modem problems" (page 579) for information on solving common modem problems.

Errors when testing the connection with cu

This section describes messages that the system displays 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 each time the kernel is relinked, a new *inittab* file is created, to retain the changes in *inittab*, you must also edit */etc/conf/init.d/sio*.

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 the "SYSTEM NOT IN Systems FILE" error 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 ⟨Return⟩.

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 "Modem answers, but terminal displays garbage" (page 581) 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.

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
```

The following 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 sitename 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.

Debugging UUCP communications

If you are unable to contact a particular machine, use the following procedure to debug the communications to that machine with **uucp** and **uutry**.

1. Log in as *root*.
2. Enter the following commands, pressing ⟨Return⟩ after each line:

   uucp -r ***testfile machine!/usr/tmp/testfile***
   /usr/lib/uucp/uutry -r ***machine***

   where ***testfile*** is a file that you want to transfer and ***machine*** is the sitename of the machine that you are trying to contact. The **-r** option overrides the retry time specified in */usr/lib/uucp/Systems*.

   NOTE If you are using **csh**, you must escape the "!" (\!). Use of **csh** for the *root* account is not supported.

3. When you see one of the following messages, press the ⟨Del⟩ key:

   ```
   Conversation Complete: Status SUCCEEDED
   Conversation Complete: Status FAILED
   ```

If your system displays the first message, check the remote system, ***machine*** for the file */usr/tmp/testfile*. If the file exists and is not empty, the UUCP connection functions. If the system displays the second message, the connection transfer failed.

If the UUCP system is set up correctly, the system attempts to transfer the *testfile* file to the remote system, ***machine***. The **-r** option to **uucp** queues the file without starting the transfer. This allows **uutry** to invoke **uucico** with debugging.

NOTE The **uutry** command displays more debugging information if you run it as *root*.

The **uutry** command saves the debugging output to the file */tmp/machine* on the local machine and prints the debugging output to your terminal using **tail -f**. Copy the output from */tmp/machine* if you wish to save it. You can change the debugging level from the default level (5) using the **-x** option.

If the system displays the "Status FAILED" message, you can look through this file for status and error messages.

See "UUCP error messages" (page 369) for an explanation of some common UUCP messages.

If you still cannot solve the problem, you may need to call support personnel. Save the debugging output for diagnosing the problem.

Alarms in UUCP audit output, data is not transferring

When sending data, UUCP expects a certain packet size and then a 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** 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 because UUCP has its own error checking.

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 describe 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** thinks 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 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 "Modem problems" (page 579) 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 thinks 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 "Modem problems" (page 579) 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. Verify that the system name is in the *Systems* file using the **uuname** command.

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:

l /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 606) for information on changing the permissions.

WRONG TIME TO CALL

The */usr/lib/uucp/Systems* file may restrict outgoing calls at this time.

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 super user, 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 following commands:

```
cd /
fixperm -n -v -dUUCP /etc/perms/*
```

This **fixperm** command displays a list of any UUCP distribution files with 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 sitename 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 as follows:

```
uname
```

To change the sitename, see Chapter 14 (page 375) in this 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 as described in this section.

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 greater than seven characters long.

NOTE **MYNAME** affects only those systems calling into the system whose *Permissions* file has been modified as described here. Calling out to a system with a system name greater than 7 characters results in that system name being truncated to 7 characters. This is only a problem if your system calls more than one system whose names are not unique to 7 characters.

UUCP is abnormally slow

Because UUCP is a batch 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 "Out of space on filesystem" (page 565) 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.

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

Table 20-1 lists several commands you can use to check for basic communications information.

Table 20-1 UUCP troubleshooting tools

Command	Description
uucheck	Allows you to check 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	Display 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).

Appendices

Appendix A
Customizing 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 directory.

The initialization files are ordinary text files and can be modified using a text editor such as **vi**(C) (see the *User's Guide*). Note, however, that 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 **init**(M).) The scripts in */etc/rc2.d* and the *.profile* and *.login* files contain commands and comments. These are in the command file format described in the chapter of the *User's Guide* entitled "The shell."

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 describes the fields in *inittab*.

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. 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 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 (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 **inittab**(F) 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. The scripts that **rc2** runs are executed in alphabetic order to ensure proper initialization.

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 gives a brief description of the scripts in */etc/rc2.d* in the order in which **rc2** executes them.

Table A-5 /etc/rc2.d scripts

Script	Description
S00SYSINIT	starts kernel message logger
S01MOUNTFSYS	mounts filesystems specified in */etc/default/filesys*
S03RECOVERY	tidies up **vi** editing sessions after a crash
S04CLEAN	removes temporary files
S05RMTMPFILES	removes temporary files
S15HWDNLOAD	downloads hardware
S16KERNINIT	starts, process accounting, network, and other kernel initialization
S20sysetup	configures print system and generates */etc/systemid*
S21perf	system accounting
S70uucp	cleans up UUCP lock files
S75cron	starts **cron** daemon
S80lp	starts **lpsched** and net utilities
S86mmdf	starts **mmdf** deliver daemons
S87USRDAEMON	starts user daemons
S88USRDEFINE	executes user-definable commands after boot
S90RESERVED	mails **fsck** output saved during autoboot to *root*

The */etc/rc2.d* directory on your system may contain scripts other than the ones listed in the previous table. The reason for this is that, 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" 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" or "K" are ignored.
- Files are executed in ASCII ascending sort-sequence order; the number in the filename determines the order of execution. In other words, the scripts are executed in the order in which they appear when you execute the **ls** command in */etc/rc2.d*.

Table A-6 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 **S01MOUNTFSYS** 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, follow these steps:

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 S (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, **S03RECOVERY** handles crash recovery and is executed after the script that begins with "S02" and before the script that begins with "S04". You must follow this naming convention to ensure that your script is executed at the proper time.
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 *S70uucp*. You can also edit any script to tailor it to your needs. For example, to start process accounting, remove the appropriate comments from the *S16KERNINIT* 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 with this form:

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*. (The backslash may be used to break a long command line; it is optional.)

```
su mmdf -c "/usr/mmdf/bin/deliver -b -T3600 \
            -cuucp -L/usr/mmdf/log/uucp.log"
```

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 **sd**(ADM) 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 apply.)

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 the chapters about the shells in the *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*, contains the greeting displayed whenever a user logs in. Initially, this file contains the name and version number of the operating system. */etc/motd* is an ordinary text file, so the system administrator can change the message by editing the file with a text editor. 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. When you announce an unscheduled system shutdown with **wall**, you should edit *motd* to include the shutdown information, so that users who log in after you run **wall** are notified of the impending shutdown.

Appendix B

Authorizing the use of job scheduling commands

This appendix describes how to permit or deny users access to use the job scheduling commands. The **sysadmsh** selection Jobs ➪ Authorize contains the authorization functions. In addition, the **at** and **batch** commands can be regulated by creation of a prototype file that specifies the environment in which the **at** or **batch** commands execute. The commands themselves are described in the chapter entitled "Managing Processes" in the *User's Guide*.

NOTE In addition to granting permission to use these commands, you must make sure that the users have the **chmodsugid** kernel authorization. If you have restricted this authorization, you must grant it as described in the chapter entitled "Administering user accounts."

Changing default authorization 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.

Changing cron defaults

To change the system default for **cron**, select the following from **sysadmsh**:

Jobs ➪ Authorize ➪ Scheduled ➪ Default

The following three selections are displayed:

None Execution is not permitted for any users.

Allow Allow all users to execute the **cron** command.

Deny Deny all users access to the **cron** command.

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

Changing the at or batch defaults

To change the system default for the **at** and **batch** commands, select the following from **sysadmsh**:

Jobs ➪ Authorize ➪ Delayed ➪ Default

The following three selections are displayed:

None Execution is not permitted for any users.

Allow Allow all users to execute the **at** or **batch** command.

Deny Deny all users access to the **at** or **batch** command.

The currently set behavior is highlighted. Use the arrow keys to highlight the behavior desired, or select the first letter. Remember that users can be allowed or denied on an individual basis as well (this is described later). The settings for individuals take precedence over the system defaults.

Allowing or denying individuals the use of cron

To change the default for **cron** for a particular user, select the following from **sysadmsh**:

Jobs ➪ Authorize ➪ Scheduled ➪ 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, the following selections are displayed:

Allow Allow this user to execute the **cron** command.

Deny Deny this user access to the **cron** command.

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

Viewing user cron permissions

To examine a list of users permitted or denied usage of **cron**, select the following from **sysadmsh**:

Jobs ➪ Authorize ➪ Scheduled ➪ View

If the system default is Allow, a list of users denied access is displayed. If the system default is Deny, a list of users allowed access is displayed.

Allowing or denying individuals the use of at or batch

To change the default for **at** or **batch** for a particular user, select the following from **sysadmsh**:

Jobs ➪ 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, the following selections are displayed:

Allow — Allow this user to execute the **at** or **batch** command.

Deny — Deny this user access to the **at** or **batch** command.

Use the arrow keys to highlight the behavior desired. This setting overrides the system default.

Viewing user at or batch permissions

To examine a list of users permitted or denied usage of **at** or **batch**, select the following from **sysadmsh**:

Jobs ➪ Authorize ➪ Delayed ➪ View

As with **cron**, if the system default is Allow, a list of persons denied access is displayed. If the system default is Deny, a list of users allowed access is displayed.

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 the following **sysadmsh** selections:

Jobs ➪ Authorize ➪ Environment ➪ At

Jobs ➪ Authorize ➪ Environment ➪ Batch

NOTE Only *root* can use these selections.

These options edit the files */usr/lib/cron/.proto* (**at**) or */usr/lib/cron/.proto.b* (**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 This is the current directory of the user at the time of submission.

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

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

$t This is the time (in seconds past January 1, 1970) that the script is run.

$< This is 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 super user can edit these files.

Example of environment file usage

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.

Appendix C

Using the system console and color displays

This appendix is concerned with utilities and features that affect the use of the system console and other color displays. Console displays are connected to a standard display adapter, while color terminals (such as Sun River terminals) are connected to the system by special adapters.

This chapter explains how to do the following:

- Set or change the console keyboard type (XT or AT) using **kbmode**(ADM).
- Protect the console from excessive wear by blanking the screen when not in use.
- Use the **multiscreen**(M) facility to control multiple screens from a single display.
- Change the font used on the screen display using **vidi**(C).
- Set the colors displayed on color screens using **setcolor**(C).

If you wish to set up a serial console, refer to "Setting up a serial console" in the chapter entitled "Administering serial terminals."

Console keyboard type selection

The operating system supports 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. Therefore if you have an AT keyboard you should reset the keyboard mode to AT, in order to make full use of the extended keyset. The **kbmode**(ADM) utility is used to test and set the keyboard mode.

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 as follows:

kbmode test

A sample session with **kbmode** in test mode is shown below, complete with user input in boldface:

```
# 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 and PC/XT mode, respectively:

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**, run the **configure**(ADM) utility and select option 13, "Hardware Dependent Parameters." Change the parameter value and relink the kernel as described in "Reallocating kernel resources with configure" in the chapter entitled "Tuning system performance."

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. (This is similar to a feature available with most terminals.) The kernel parameter **TBLNK** controls the VGA console screen protection feature. By default, screen blanking is not performed: to enable this feature, you must invoke the **sysadmsh** selection

System ➪ Configure ➪ Kernel ➪ Parameters.

Select category 6: "MultiScreens", and change the value of **TBLNK** to the number of seconds that the system should wait before blanking the screen. The kernel must then be relinked and booted for the new behavior to take effect. Use the **sysadmsh** System ➪ Configure ➪ Kernel ➪ Rebuild selection to relink the kernel.

Using MultiScreen

With multiscreen, you can use your console as several terminals at one time. Pressing a simple key combination switches you from one screen to another, with each screen acting as an independent terminal.

Each multiscreen is independent, which means that you can log in and run programs on each screen. Output from your programs is saved in a screen buffer, so you see the most recent output for whichever screen you look at. If you stop output to one screen, as when you press the ⟨Ctrl⟩**s** key combination, only that screen is affected.

The amount of memory in your computer determines the number of multiscreens available on your system. When you boot your system, the number of automatically-enabled multiscreens is displayed. Most machines have between two and six multiscreens enabled, but your machine can have up to twelve if your system has sufficient memory. To increase the number of multiscreens on your system, you need to add to your system's memory; additional screens will be enabled automatically.

Although all of the multiscreens can be open and active at once, you see only one screen at a time. The selected multiscreen is like a terminal that is "connected" to the keyboard. Switching between screens is like moving to another terminal because each multiscreen has its own device file.

The multiple screen feature uses the */dev/tty[01...12]* device files. These files provide character I/O between your system and your computer screen and keyboard.

To select any active screen, press ⟨Alt⟩⟨F*n*⟩, where ⟨F*n*⟩ is one of the function keys on your keyboard. Function keys are generally located across the top or down the far left side of the keyboard. The *tty01* is the ⟨Alt⟩⟨F1⟩ terminal, *tty02* is the ⟨Alt⟩⟨F2⟩ terminal, *tty03* is ⟨Alt⟩⟨F3⟩, etc. For example, the following keystroke switches you to screen 6, corresponding to */dev/tty06*:

⟨Alt⟩⟨F6⟩

You can also rotate through the screens by pressing the Control and Print Screen key combination, ⟨Ctrl⟩⟨PrtSc⟩ (using the ⟨Ctrl⟩ key and the ⟨PrtSc⟩ key). Use this combination to access screens for which you do not have function keys. For example, if you have twelve multiscreens enabled, but your computer keyboard has only ten function keys, display screen eleven by pressing ⟨Alt⟩⟨F10⟩ to get to screen 10, and then pressing ⟨Ctrl⟩⟨PrtScr⟩ to rotate to screen 11. To access screen 12, press ⟨Ctrl⟩⟨PrtScr⟩ again. Pressing ⟨Ctrl⟩⟨PrtScr⟩ again rotates you back to the first multiscreen, *tty01*.

Note that you can use ⟨Ctrl⟩⟨Alt⟩***function-key*** combinations in addition to ⟨Alt⟩***function-key*** combinations to change multiscreens. This is especially useful in applications that reserve the ⟨Alt⟩***function-key*** combinations for their own use. This can be configured using the **mapkey**(ADM) utility.

For more information, refer to **multiscreen**(M) and **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 enter the **sysadmsh**(ADM).
2. Use the System ➪ Configure ➪ Kernel ➪ Parameters selection and select category 6, Multiscreens.
3. Skip the parameters displayed by pressing ⟨Return⟩ until you reach the **NSCRN** parameter. Enter a value corresponding to the number of screens you wish to enable.
4. Calculate the amount of screen memory in Kbytes (controlled by the **SCRNMEM** parameter) as follows:

   For 25-line displays: **SCRNMEM** = 10K + 4K * **NSCRN**

   For 43-line displays: **SCRNMEM** = 10K + 8K * **NSCRN**

5. Enter a value for **SCRNMEM** to match the value you calculated in the previous step.

6. Exit the **configure**(ADM) menu by entering " q " and pressing ⟨Return⟩.
7. Use the System ➪ Configure ➪ Kernel ➪ Rebuild selection to build the new kernel with the revised screen values. (The Rebuild should be visible when you exit the previous selection.) Follow the prompts. Exit **sysadmsh** when the process is complete.
8. You must then 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

9. You should now shut down the system and reboot from the new kernel. Enter the following command:

   shutdown -g0

   When the reboot message is displayed, press ⟨Return⟩ 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

Changing 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 C.1 lists the font definition files in the directory */usr/lib/vidi*.

Table C-1 Font definition files

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 color displays with setcolor

setcolor(C) is a simple utility that enables you to control the colors used on the display screen. (The **setcolor** command usually has no effect on monochrome displays or terminals.) 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 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 following flags are available. In the arguments below, "color" is taken from the above list.

Changing the foreground and background colors

You can set both background and foreground colors with a single command as in the following example:

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

Reverse video normally inverts the foreground and background colors. **setcolor** allows you to set these 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. For example, 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

The **-n** option returns the screen to "normal" white characters on black background.

Appendix D

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 and terminals. For a full description of the special files mentioned here, see the manual pages in the Hardware Dependent (HW) section.

UNIX directories

The following subsections discuss each of the main directories of the operating system.

The root directory

The root directory (/) contains the following system directories:

/bin UNIX command directory

/dev device special directory

/etc additional program and data file directory

/lib C programming library directory

/mnt mount directory (reserved for mounted filesystems)

/usr user service routines (may contain user home directories)

/tcb system files that are part of the TCB (Trusted Computing Base)

/tmp temporary directory (reserved for temporary files created by programs)

All of the above directories are required for system operation.

The *root* directory also contains a few ordinary files. Of these files, the most notable is the */unix* file, which contains the UNIX kernel image.

The /bin directory

The */bin* directory contains the most common UNIX commands, that is, the commands likely to be used by anyone on the system.

The /dev directory

The */dev* directory contains special device files that control access to peripheral devices. All files in this directory are required, and must not be removed.

There are several subdirectories to the */dev* directory. Each of these subdirectories holds special device files related to a certain type of device. For example, the */dev/dsk* directory contains device files for floppy and hard disks. The operating system supports both XENIX and UNIX device naming conventions. Where appropriate, the files in the */dev/dsk* directories are linked to the device files that exist in */dev*. You can access the same device through the file in */dev* or the file for the same device in a subdirectory of */dev*.

Table D-1 contains a list of the more common devices.

Table D-1 Commonly-used /dev device nodes

UNIX device	XENIX device	Name
/dev/console	same	system console
/dev/rdsk/*	/dev/r*	raw devices
/dev/dsk/0s0	/dev/hd00	entire disk on drive 0
/dev/dsk/0s1	/dev/hd01	first disk partition on drive 0
/dev/dsk/0s2	/dev/hd02	second disk partition on drive 0
/dev/dsk/1s0	/dev/hd10	entire disk on drive 1
/dev/dsk/1s1	/dev/hd11	first disk partition on drive 1
/dev/dsk/1s2	/dev/hd12	second disk partition on drive 1
/dev/dsk/f05d9	/dev/fd048ds9	360K floppy drive 0

(Continued on next page)

Table D-1 Commonly-used /dev device nodes
(Continued)

UNIX device	XENIX device	Name
/dev/dsk/f05q	/dev/fd096ds9	720K floppy drive 0
/dev/dsk/f05h	/dev/fd096ds15	1.2MB floppy drive 0
/dev/dsk/f03h	/dev/fd0135ds18	1.44MB floppy drive 0
/dev/lp	same	lineprinter
/dev/kmem	same	kernel virtual memory
/dev/mem	same	physical memory
/dev/null	same	null device
/dev/rmt0	/dev/rct0	default cartridge tape device
–	/dev/rft0	QIC-40 tape device
–	/dev/rctmini	minicartridge tape device
/dev/root	same	root file structure
/dev/swap	same	swap area
/dev/ptyp***nn***	–	pseudo tty (master)
/dev/ttyp***nn***	–	pseudo tty (slave)
/dev/tty***nn***	–	console multiscreen

The /etc directory

The */etc* directory contains miscellaneous system program and data files. All files are required, but many can be modified. 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 623) for more information on the */etc/rc* directories.)

The data files in the directory */etc/default* contain default information that is used by system commands (see **default**(F)). The data files shown in D-2 may be modified, but may not be removed.

Table D-2 /etc/default files

Filename	Utility
/etc/default/archive	**sysadmsh**(ADM) backup default information
/etc/default/authsh	**sysadmsh**(ADM) default account information
/etc/default/backup	**backup**(ADM) default information
/etc/default/boot	**boot**(ADM) information
/etc/default/cleantmp	**cleantmp**(ADM) default information

(Continued on next page)

Table D-2 /etc/default files
(Continued)

Filename	Utility
/etc/default/cron	**cron**(C) default logging information
/etc/default/device.tab	**pkgadd**(ADM) default information
/etc/default/dumpdir	**xdumpdir**(ADM) default information
/etc/default/filesys	**sysadmsh**(ADM) default filesystem data
/etc/default/format	**format**(C) default information
/etc/default/goodpw	**goodpw**(ADM) default password check data
/etc/default/idleout	**idleout**(ADM) default information
/etc/default/lang	default locale information
/etc/default/lock	**lock**(C) default information
/etc/default/login	**login**(M) default information
/etc/default/lpd	**lp**(C) default information
/etc/default/man	**man**(C) online man page default information
/etc/default/mapchan	**mapchan**(M) default information
/etc/default/mapkey	**mapkey**(M) default information
/etc/default/micnet	**micnet**(C) default information
/etc/default/msdos	location of DOS disks (A:, B:,...) for **dos**(C) commands
/etc/default/passwd	**passwd**(C) default information
/etc/default/purge	**purge**(C) default information
/etc/default/restor	**xrestore**(ADM) default information
/etc/default/scsihas	defines SCSI host adaptors
/etc/default/slot	defines EISA slots (EISA machines only)
/etc/default/su	**su**(C) default information
/etc/default/tape	**tape**(C) default device information
/etc/default/tar	**tar**(C) default device information
/etc/default/usemouse	**usemouse**(C) default information

The /lib directory

The */lib* directory contains runtime library files for C and other language programs. This directory is required.

The /mnt directory

The */mnt* directory is an empty directory reserved for mounting removable filesystems.

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* file contains a record of all error messages sent to the system console. This file is especially useful for locating hardware problems. For example, an unusual number of disk errors on a drive indicates a defective or misaligned drive. Because messages in the file can accumulate rapidly, the file must be deleted periodically.

The /tcb directory

The */tcb* directory contains all files that are part of the TCB (Trusted Computing Base). These files comprise the security enhancements made to the operating system to make it more secure than other UNIX operating systems. The TCB files and their formats are discussed in the **authcap**(F) manual page. For a discussion of how these files extend the traditional UNIX security database files, see "How account information is stored" (page 58). For a complete discussion of system security, see the "Maintaining system security" chapter.

The /tmp directory

The */tmp* directory contains temporary files created by UNIX programs. The files are normally present when the corresponding program is running, but may also be left in the directory if the program is prematurely stopped. You can remove any temporary file that does not belong to a running program.

Log files

A variety of directories contain log files that grow in size during the normal course of system operation. Many of these files must be periodically cleared to prevent them from taking up valuable disk space. (See the section on "Checking and clearing log files" in the "Managing filesystems" chapter for more information.) The following lists the system log files (by full pathname) and their contents:

/etc/ddate
: records date of each backup.

/usr/adm/pacct
: records accounting information; grows rapidly when process accounting is on. (See **accton**(ADM) and **acctcom**(ADM).)

/usr/adm/messages
: records error messages generated by the system when started.

/etc/wtmp
: records user logins and logouts. (See **login**(M).)

/usr/adm/sulog
: records each use of the **su** command; grows only if option is set in the */etc/default/su* file. You must create */etc/default/su*. (See **su**(C).)

/usr/lib/cron/cronlog
: records each use of the **at**(C) and **cron**(C) commands.

*/usr/spool/uucp/.Log/utility/sitename/**
: logs UUCP commands used over a UUCP network. The ***utility*** and ***sitename*** are the name of the UUCP utility and the name of the remote site, respectively.

*/usr/spool/uucp/.Log/.Old/**
: stores old log files placed in this directory by the **uudemon.clean** shell script.

Special device files

Many of the filesystem maintenance tasks described in this guide require the use of special filenames, block sizes, and gap and block numbers. The following sections describe each in detail.

Special filenames

A special filename is the name of either the device special block or character I/O file, which corresponds to a peripheral device such as a hard or floppy disk drive. These names are required in such commands as **mkfs**(ADM), **mount**(ADM), and **df**(C) to specify the device containing the filesystem to be created, mounted, or searched.

Table D-3 lists the XENIX and UNIX special filenames and corresponding devices for hard and floppy disk drives on a typical computer.

Table D-3 Disk device filenames

Filename	Disk drive
/dev/fd0	floppy drive 0
/dev/dsk/f0	floppy drive 0
/dev/fd1	floppy drive 1
/dev/dsk/f1	floppy drive 1
/dev/hd00	entire hard disk
/dev/dsk/0s0	entire hard disk
/dev/root	root filesystem
/dev/u	user filesystem

For a complete discussion of disk device files, see the **hd**(HW) and **fd**(HW) manual pages.

Block sizes

The block size of a disk is the number of blocks of storage space available on the disk, where a block is 1024 bytes of storage. Most commands, however, report disk space in terms of 512 byte blocks, in particular **df**(C), **du**(C), **ls**(C), **lc**(C), and **find**(C). A 500-byte file on a 1024-byte block filesystem is reported as using 2 blocks by these utilities, as the file uses one physical disk block that is equivalent to two 512-byte filesystem blocks. The size of a 40-megabyte hard disk in 1024-byte blocks is 39168. Note that some of the blocks on the disk are reserved for system use and cannot be accessed by user programs. The block size of a typical floppy disk depends on the total storage capacity of the disk, as given by the manufacturer.

Gap and block numbers

The gap and block numbers are used by the **mkfs**(ADM) and **fsck**(ADM), commands to describe how the blocks are to be arranged on a disk. Table D-4 lists the gap and block numbers for the floppy and hard disks used with a typical computer.

Table D-4 Gap and block numbers

Disks	Gap	Block
floppy disk, 48ds9	1	9
floppy disk, 96ds15	1	15
floppy disk, 135ds9	1	9
floppy disk, 135ds18	1	18
hard disk	1	34

The number of blocks can also be determined by multiplying the number of sectors per track (for example, 17) by the number of heads on the hard disk, dividing by 2 (because there are 2 sectors per block), and rounding down to the nearest integer.

Serial devices

The **enable**(C) and **disable**(C) commands enable and disable logins on terminals. **enable** and **disable** require the names of the serial lines through which a terminal or network is to be connected. Table D-5 lists the device special filenames of the two serial lines (actually two serial ports either with or without modem control). The character I/O files corresponding to these serial lines can be found in the */dev* directory. Note that the files */dev/console* and */dev/tty01* through */dev/tty12* represent "hardwired" devices and are not available for connection to terminals or hardware. Also, refer to **serial**(HW) for more information on serial lines.

Table D-5 Serial devices

Filename	Line
/dev/tty1a	main serial line (without modem control)
/dev/tty2a	alternate serial line (without modem control)
/dev/tty1A	main serial line (with modem control)
/dev/tty2A	alternate serial line (with modem control)

Index

Special characters

@ (at symbol), 290
/ (slash), root filesystem, 95

A

Absolute pathnames, 179
accept(ADM), 148
Acceptance status, printer, 150
Access time, locating files by, 105
Account
  activity reporting, 220-222, 224-225
  anonymous, 205
  copying from other systems, 80
  duplication on other machines, 79, 80
  enabling, 596
  locking, 57, 68, 89, 576
  modifying, 64-84
  moving, 78
  password, 576
  sysadmsh(ADM) options, 38
  transitions with su(C), 78
  user, 576
"Account is disabled -- see Account Administrator", 576, 596
"Account is disabled but console login is allowed", 595-596
Accountability, 203
Accounts administrator
  password unlocking, 210
  responsibilities, 205
Acer Fast Filesystem (AFS), 99
  NMPBUF parameter, 534
  NMPHEADBUF parameter, 534
  repairing with fsdb(ADM), 574
Action field, /etc/inittab file, 620
Activity reports, 220
ACU dialer type, 348-349
Adding
  administrator, 66
  computer, 14
  login shell, 94
  media defaults, 182
Adding *(continued)*
  printer, 121
  tape drive, 36
  user, 57, 60-64
Address, mail, incorrect, 577
addxusers(ADM), 80
ADM reference section, 8
AF_INET, 357
AGEINTERVAL parameter, 537
"aio_breakup ... buffer table overflow", 494
"aio_breakup ... request table overflow", 494
"aio_memlock ... process table overflow", 494
"aio_setlockauth ... lock table overflow", 494
Alerting
  mount print wheel, 130-131
  printer faults, 131-133
Alias file, 412-415, 446-447
ALL value, COMMANDS option, 345
"allocreg - Region table overflow", 494, 536
Allow list for printer forms, 134
Allow option, 630-632
Alt key, 559
Anonymous account, 205
ANSI terminals, 295
ap(ADM), 80
Applications, sysadmsh(ADM) selection, 37
Archiving, 175
  directories, 178
  floppy disk, 181, 183
  sysadmsh(ADM) options, 41, 42
  tape, 178
ArpaNet. *See* Internet
asroot(ADM), 77
asroot(ADM), command, 77
ASSERT error messages, 370
Asynchronous I/O (AIO), 548
AT console keyboard, 560, 634
At symbol (@), 290
"at: you are not authorized to use at...", 562
at(C), 630-632
  sysadmsh(ADM) option, 41
  troubleshooting, 561-562
AT&T 5310/20 matrix printer, 124
Attributes, printer, 135-136

Audit, 205, 212
  administration, 93, 205, 209, 250
  analysis, 241
  authorizations, 91, 209, 242
  backups, 262-263
  collection, 251, 263, 286
  configuring, 257
  crash behavior, 250
  daemon, 240, 258, 286
  database, 228
  device driver, 239
  directories, 260, 263, 265-266
  disk space, 250, 263
  displaying status, 255
  enabling/disabling, 255
  event masks, 246
  event types, 244, 253
  files, 260-261, 263
  frequency, 257
  goals, 248-249
  listing audit directories, 264
  mandatory, 245
  mask, 251
  records, 242, 278, 283-285
  reduction, 287
  report generation, 266, 278
  report template, 266-267, 274-276
  session, 286
  subsystem, 93, 237-238, 250, 596
  suspending, 288
  sysadmsh(ADM) options, 37, 75-76, 241
  trail, 237, 256
  user accounts, 75-76
"Audit: filesystem is getting full", 596
audittrail authorization, 92, 242
auth authorization, 91, 209
authcap(F), file format, 58
authck(ADM), 50, 58, 61, 231, 488
Authentication
  administration, 209
  database, 228-229, 231
  trusted system, 204
"Authentication database contains an inconsistency", 597
"Authentication error; see Account Administrator", 597
AUTHLOG, 449
Authorization, 204
  assigning, 38, 74-75
  changing, 90
Authorization, 204 *(continued)*
  default, 81, 83
  kernel, 91-93, 208, 209
  mount filesystem, 98
  primary, 92
  privileges, sysadmsh(ADM), 207
  problems, 599
  secondary, 92
  shutdown, 92
  su, 78
  subsystem, 91-92, 208
  sysadmsh(ADM) options, 41
authsh(ADM), 67, 78
Auto-answer modem configuration, 313
autoboot(ADM), 56
Autologic APS-5 phototypesetter, 124
Autologout. *See* idleout(ADM)
Automatic Call Unit (ACU), 311, 327, 337, 348-350, 605
Automatic Write Remapping (AWRE), 563
AUX port, 138

B

Background UUCP daemons, 327
Backspace key, 9
Backspace (stty echoe), 294
Backups, 453-478
  audit files, 109, 262
  authorization, 91, 209
  creating, 567
  levels, 474-477
  listing, 469-471
  media security, 202
  restoring, 38, 471-474, 567
  scheduled, 456-465
  sysadmsh(ADM), 38, 454-456
  troubleshooting, 482
  unattended, 477-478
  unscheduled, 465-468
  verifying, 468-469
Bad
  block, 119
  octal digit, 599
  system name, 608
Bad track
  boot problems, 551
  mapping with badtrk(ADM), 562
badhosts channel, 380, 423

badusers channel, 380, 423
Base address, tape drive conflict, 599
batch(C), 561-562, 630-632
Baud rate, 293
  error, 331
  modem, 311
  modem problems, 580
  printer, 160-161, 166
  setting in /etc/inittab file, 584
  terminal, 290, 291
bcheckrc(ADM), 557
bdflush daemon, 530
BDFLUSHR parameter, 530
BFREEMIN parameter, 534
/bin directory, 642
/bin/login file, restoring, 555
"/bin/su: cannot setgid to auth, no auth entry", 576
/bin/sulogin file, restoring, 557
Block, 103, 115
  bad, 119
  device activity, sar -d, 515
  displaying usage, 104-105
  duplicate, 119
  free-block list, 119
  numbers, 648
  size, 647
  structure, 115
Boot
  automatic, 56
  configuring, 55-56, 619-627
  display, 54
  filesystems, 36, 97
  load extension, kernel parameters, 548
  restoring files, 51
  security, 229, 488
  steps, 43-45, 47
  troubleshooting, 551-558
  using old kernel, 55
Boot floppy, 552
  Emergency Boot Floppy Set, 553-554, 558
  restoring root filesystem, 568
"/boot not found", 552
Boot screen
  port does not respond, 590
  tape drive recognition, 599
boot(HW), 55
Bootstrap program, 43, 55
Bootstring options, 55
bootwait action, /etc/inittab file, 621
Bourne shell
  .profile file, 296
  TERM assignment, 295
Break, login script, 339
BREAK, 290
Buffer, 116, 525
  activity, sar -b, 513
  allocating character list, 531
  allocating physical I/O buffers, 528
  buffer cache, 529
  increasing cache hit rate, 528
  parameters, 528
  specifying age for filesystem updates, 529
  tape size, 529
  transfer, 529
  writing to disk, 530

C

C program files, 644-645
C reference section, 8
C. (work) files, 326, 329, 367-371
C2 security, 12, 83, 201-210
Cabling
  planning, 14
  unterminated wire problem, 309
Cache hits, reducing disk accesses, 528
CALLBACK option, Permissions file, 344
Call-out table, 538
Cancelling print request, 143
"CANNOT ACCESS DEVICE", 606
"Cannot create", 589
"Cannot load floating point emulator", 554
"Cannot obtain database information on this terminal", 575, 598
"cannot open: device busy", 306
"Cannot open /dev/rct0", 601
"Cannot open /etc/inittab", 556
"Cannot read emulator file header", 554
"Cannot rewrite terminal control entry for tty01", 576
"can't change mode of job", 562
"Can't rewrite terminal control entry for tty", 597
Carrier Detection (CD) light, modem problems, 581
Carrier Detection (CD) line, modem configuration, 313
Cartridge tape. *See* Backups

cbackup(ADM), 477
Changing password, 57
Channel, 375, 422
Channel files, 418
Character buffers parameters, 531
Character set, printer, 128
Chat script. *See* Login, script
checkaddr(ADM), 435, 577
Checking
  auth database, message, 47
  filesystems, 114
  network, 326
  protected subsystems database, message, 47
  tcb, message, 47
  ttys database, message, 47
checkque(ADM), 435
checkup(ADM), 434
chmod(C)
  finding files by permissions, 107
  removing sticky bit, 575
  setting directory SGID, 100
chmodsugid
  at(C), 562
  kernel authorization, 92, 208, 212, 629
ERROR: UNCLOSED RANGE ON PREVIOUS ENTRY
chown authorization, 92
Class, printer, 122, 144-145, 147
Cleaning, filesystem, 44
"CLEANING NON SYSTEM 3 FILESYSTEM", 568
cleanque(ADM), 436
cleantmp(ADM), 102
Clearing, bits, 213
Cleartext, 60
Clock
  parameters, 538
  setting at startup, 46
CMASK parameter, 533
cmp(C), 472
Collection directories, 263
Color displays
  changing display, 638
  changing foreground and background, 638
  changing screen border, 639
  list of display, 638
  reverse video, 639
  using, 633
Colors, under sysadmsh(ADM), 35
COLUMNS setting, 302
COM domain name, 386
COM ports
  cabling, 309
  dialing in, 314
  interrupts, 312
  modem use, 309
Commands, 7, 9, 642, 645
  default remote, 345
  executing remote, 323
  printer, 136, 137
  UNIX, 325
  UUCP, 325, 326
COMMANDS option, Permissions file, 344, 345
Compacting files, 42
Comparing files, 42
compress(C), 103
compressed, printer type, 135, 165
Computer,setting up, 14
"CONFIG: aio_breakup - AIO buffer table overflow ...", 548
"CONFIG: aio_breakup - AIO request table overflow ...", 548
"CONFIG: aio_memlock - AIO process table overflow ...", 548
"CONFIG: aio_setlockauth - AIO lock table overflow ...", 548
configaudit authorization, 92, 208, 242, 287
Configuration file, UUCP, 327
configure(ADM), 504-505
Configuring
  modem, 306
  print service, 157-159, 163-171
  printer, 122-124, 154-155, 160-163, 172
  user accounts, 64-76, 81, 83
  UUCP, 324, 332-336
"Connect failed: NO DEVICES AVAILABLE", 580
Connecting
  modem, 307, 330
  printer, 127, 171
  systems by wire, 360-361
  UUCP, 328
Console, 633
  See also System
  changing display colors, 638-639
  device file, 648
  device name, 643
  keyboard type, selecting, 634
  multiscreens, 597, 635
  resetting the screen, 639

Console, 633 *(continued)*
screen protection, 635
serial, 297
troubleshooting, 558-561, 575
Content types, 125
Context indicator (sysadmsh), 19
Control
map size, specifying, 540
Register 3 (CR3), 546
Control keys, 9
q, unlock keyboard, 559
s, stop scrolling, 559
Converting, XENIX to MMDF, 428
Copying
file, 41
floppy disks, 182-183
remote files, 326
Core file
creating, 10
recursive removal, 108
Corruption
data, 227
filesystem, 114
repairing, 118
CP reference section, 8
cpio(C)
backup, 455, 472, 477
file ownership, 216
cps(ADM), 50
CPU usage, 512, 519
Crash recovery, 229, 488
crash(ADM)
analyzing core image, 505
reading putbuf buffer, 530
STREAMS display headings (table), 507
studying memory dump, 482
Creating
dial-in accounts, 332
Emergency Boot Floppy Set, 552
crnlmap printer, 124
"cron may not be running", 561
cron(C)
authorization, 91, 209, 629-631
clearing log files, 110
daemon, 233, 477, 561
reducing system load, 511
troubleshooting, 561-562
UUCP, 326, 328, 366
cronlog log file, 646
crontab(C), 477
crontab(C) directory, 512, 520
crypt(C), 215
CS8, example, 291
"ct: ERROR: Tape controller ... not found", 599
CTBUFSIZE parameter, 529
ct(C), 325
"cu: dir permission denied", 311
cu(C)
dialing out, 313, 315
executing, 325
login sequence, 339
modem connection errors, 606-607
print debugging information, 605
testing a modem, 311
troubleshooting modem, 579
Current logins, sysadmsh(ADM) selection, 36
custom(ADM), restoring files, 50

D

D. (data) files, 326, 367, 369, 371
DAC. *See* Discretionary Access Control
Daemons
audit, 258
printing, 158
sdd, 233, 626
security, 233-234
UUCP, 326-328, 345-346
Daisy wheel, 145
DARPA, 375
See also Internet
DASI terminals, 124
DAT tape drive size, 178
Data
bit settings, modem problems, 581, 584
block, 115
blocks in filesystem exhausted, 565
encryption, 215
protecting, 212
region, 536
storage, 115-116
Data Communications Equipment (DCE), modem configuration, 309
Data compression, modems, 583
Data Terminal Equipment (DTE), computer serial configuration, 309

Data Terminal Ready (DTR), 308
  auto-answer modem, 313
  modem problems, 582
Databases, account
  authentication, 228-229
  configuring, 59-60
Date, setting, 46
dbmbuild(ADM), MMDF, 434
ddate log file, 646
dd(C), 182-183
DDN, Network Information Center, 387
Debugging, UUCP communications, 609
Default
  accounts, 67, 81
  filesystems check, 36
  home directory, 36
  printer, 39, 135-136, 145
  security, 81, 83
  sysadmsh(ADM) options, 41
  UUCP, 345
Defaults, /etc/default directory, 643
DEFBOOTSTR boot option, 55
Delete key, 9
deliver daemon, fixing mail problems, 578
deliver(ADM)
  background program, 432
  checkque(ADM), 435
Delivery, channels, 422
Deny list for printer forms, 134
Deny option, 630-631
Description line (sysadmsh), 19
Destination, printer, 122
/dev directory, 642
/dev/hd0a, UNIX device name, 189
/dev/hd0root, running fsck(ADM), 568
Device, 348
  assignment database, 228
  cannot access, 606
  dialout modem problems, 580
  directories, 642, 644
  field, Systems file, 337
  filenames, 647
  filesystem, 96
  fixing file permissions, 606
  hardwired, 648
  locked, 611
  modem errors, 306
  naming, 38
  nodes, 642
  none available, 580, 607, 611
Device, 348 *(continued)*
  parameters, 545
  permissions in UUCP, 612
  printer, 122, 172
  problem in UUCP, 611
  sysadmsh(ADM) options, 38
  UUCP requirements, 328
Devices file
  baud rate, 330
  dial-out entries, 315, 348
  error, 371
  example control file, 329
  for TCP/IP, 356
  format, 348
  local area network, 338, 349
  matching Systems file, 328-329
  modem, 311, 329
  speed field, 330
  testing modem, 333
  UUCP configuration, 327
  write permissions, 311
/dev/inet/tcp, 358
/dev/lp, device, 172
/dev/syscon, 585
df(C), 103, 565, 647
Dialcodes
  file, 337, 353
  UUCP, 335
Dialer programs, 315
  binaries, 316
  /usr/lib/uucp, 329
Dialers file, 353
  Automatic Call Unit (ACU), 349
  dialer types, 315
  dialer-token field, 350-352
  error, 374
  service code example, 359
  uucico(C), 329
Dialer-token, 348, 351-352, 373
Dial-in
  modem, 310, 314
  site, 330, 332, 335
Dialing out, syntax, 313
Dial-out modem, 315
dialups file, 314
diff(C), 472
Direct keyword, 348-349
Direct line, 311

Directory
daemons, 234
finding, 105-106
GID bit, filesystem, 100
problems, cannot remove files, 575
root, 641
security, 213, 215
size, 13, 104, 111-112
sysadmsh(ADM) options, 41
disable(C), 296, 648
modem use, 309
printer, 137, 143
Disabled
account, 596
terminal, 598
Disabling
audit, 37
C2 features, 235, 547
Discretionary Access Control (DAC), 203, 237
Disk
See also Floppy disk; Hard disk
adding second, 113
audit record space, 263
bad track table, 562
blocks, 647-648
changing layout, 113
configuring, 113
device names, 643
errors, 645
filenames, 647
fragmentation, 103, 111, 567
free space, 102-103
gap number, 648
heads, 648
parameters, 528
performance, improving, 494
repairing, 114
security, 202
sysadmsh(ADM) selection, 36
troubleshooting, 562-564, 567
usage, 4, 104
diskcp(C), 182-183
Display
console, 10
sysadmsh(ADM), 20
Display adapter, displaying characters, 637
divvy(ADM), error during installation, 572
"dk_name - Diskinfo table overflow", 494, 528
DMA (Direct Memory Access) channel
allowing simultaneous requests, 546
tape drive conflict, 599-600
transferring buffers requests, 529
DMAABLEBUF parameter, 529
DMAEXCL parameter, 546
dmesg(ADM), 482
DO387CR3 parameter, 546
Domain
file, 415, 418
fully qualified, 386
levels, 385-386
name, registering, 387
DOS
coexistence with UNIX, 187
defining device names, 195
file and directory arguments, 195
files, 187, 196, 199
filesystems, 187, 196-198
floppy disks, mounting, 198
hard disk devices, 191
installing UNIX system partition, 192
partition, removing, 194
sysadmsh(ADM) options, 41-42
utilities, 187, 194-196
DOS reference section, 8
DOSINODE parameter, 535
"dosiread - DOS inode table overflow", 494
DOSMOUNT parameter, 535
d_passwd file, 314
DQP-10 matrix printer, 124
DSTFLAG parameter, 538
du(C), 104, 566
dumb, printer, 124

E

EBUSY error, 533
Echo problems
modem, 579
terminal, fixing, 604
echoe parameter, 294
ECON-80 mode, 302
Editing
command line, 9
sysadmsh(ADM) option, 41
EDU domain name, 386
80387 math coprocessor, 546
eisa(ADM), 55

elite pitch, 135, 165
Emergency Boot Floppy Set, 552
  restoring root filesystem, 568
  restoring system files, 552-554, 556-558
emulator, printer type, 124
enable(C), 648
  modem port, 313
  printer, 137, 143
Enabling
  audit, 37
  disabled account, 596
  terminal, 598
Encryption, 215
Enter key, 9
Enter new time ([yymmdd]hhmm): message, 46
env(C), 296
environ(M), 456
Environment files, 632
Environment variables
  PATH, 490
  sysadmsh(ADM), 35
  TZ (timezone), 538
Epson printer, 124
Erase key, 9
Error checking (fsdb), disabling, 574
Error codes, auditing, 243
Error Correction Code (ECC), 180
Error correction, modems, 583
Error logs, 482, 645
Escape, login script , 339
Escape key, 9
/etc directory, 643
/etc/auth/system/default file, 59
/etc/auth/system/gr_id_map file, 50
/etc/auth/system/pw_id_map file, 50
/etc/auth/system/ttys file, 597-598
/etc/bcheckrc file, 557
/etc/cleanup script, 110
/etc/conf/cf.d, 553
/etc/conf/cf.d/init.base, 620
/etc/conf/init.d/sio, 310
"/etc/conf/pack.d/ct/space.c line 46 bad octal digit", 599
/etc/cshrc file, 626
/etc/ddate file, 108-109
/etc/default directory, 55, 177, 194, 643
/etc/default/authsh file, 67
/etc/default/boot, 297
/etc/default/cleantmp file, 102, 566
/etc/default/filesys, 98
/etc/default/format file, 182
/etc/default/goodpw file, 89
/etc/default/idleout, 211
/etc/default/login file, 58
/etc/default/lpd, 171
/etc/default/msdos file, 195
/etc/default/passwd file, 58, 89
/etc/default/tar file, 177
/etc/d_passwd, 314
/etc/emulator file, 554
/etc/gettydefs file, 290-292, 584
/etc/group, 231
/etc/group file, 76-77, 362, 488, 576
/etc/inetd.conf, 355
/etc/init file, 558
/etc/inittab file
  editing, 310
  field descriptions, 620
  format, 620
  modifying system initialization, 619
  recreating, 556
  setting baud rate in, 584
  telinit(M), 620
  UUCP, 363, 365
/etc/motd file, 36, 102, 627
/etc/mscreencap, 299
/etc/passwd file, 58, 60, 80, 204, 231, 314, 362, 364, 488
/etc/profile file, 626
/etc/rc* scripts, 45, 643
/etc/rc2.d scripts, 623-624
  file execution order, 624
  S and K files, 624
  S86mmdf, 432
  S87USRDAEMON, 359
  starting cron daemon, 561
  starting deliver daemon, 578
  starting lpsched daemon, 587
  system initialization files, 619
/etc/services, 355, 357
/etc/shadow file, 58, 60
/etc/shutdown program, 117
/etc/systemid file, UUCP, 363
/etc/termcap, 294
/etc/ttytype, 296
/etc/wtmp file, 109-110
ETRUNC parameter, 100, 533
EVDEVS parameter, 545
EVDEVSPERQ parameter, 545

Event
audit, 37
mask, 246, 287
type, 287
"event - Event channel full", 494, 545
"event - Event table full", 494, 545
EVQUEUES parameter, 545
Exabyte (8mm) tape drive, size, 178
execsuid authorization, 92, 208
Executing
recursive commands, 106-107
remote programs, 330
system programs, 38
Exit codes, printer, 167
Exiting
shell, 9
sysadmsh(ADM), 42
Expanding, system, 14
Expect string, 338-339
Export example, 295
expr(C), 301
Extended Acer Fast Filesystem (EAFS), 99
EXTRA_NDEV parameter, 548
EXTRA_NEVENT parameter, 549
EXTRA_NFILSYS parameter, 549

F

F reference section, 8
Failure mode, 83
"falloc - File table overflow", 494, 533
Fault
alerting, 133
printer, 131-134
fdisk(ADM)
main menu, 188
partition, 189, 194
switching operating systems, 191
UNIX system and DOS versions, 187
File
access operations, sar -a, 513
archiving, 175, 178, 183
audit system, 109
backups. *See* Backups
cannot remove from directory, 575
checking MMDF configuration files, 578
copying remote, 328-329
corruption, 48-50, 231
DOS, 42
File *(continued)*
environment, 632
finding, 105-108
GID bit, 100
last access time, 105
locating, 566
long filenames, 99
maximum number of open, 533
modifying system initialization, 619
monitoring, 110
parameters, 532
process accounting, 109
recursive removal, 108
repairing, 118
restoring, 51, 552
See also Backups
security, 208, 215
size, 104
size limit, 536
structures, 115
symbolic links, 99
sysadmsh(ADM) options, 41
transfer, UUCP, 323, 326-329, 367
transferring ownership, 107
UUCP log files, 109
File Control database, 229
File table
allocating open entries, 533
status, 520
Filename
device special files, 647
truncation, 100
"Filename too long", 533
Filesystem, 95-96
See also File
adding, 96, 113
avoiding problems, 481
backups. *See* Backups
checking, 36, 117, 119
cleaning, 44
configuring, 100
corruption, 47-50, 114, 116, 227, 230-231
creating, 111
expanding, 113
floppy disk, 184-186
free space, 102-103, 596
maintaining, 111
mounting, 97, 216, 565
parameters, 532
removing, 111, 117

Filesystem, 95-96 *(continued)*
repairing, 117-118
restoring. *See* Backups, restoring
root, 95, 118
security, 212, 216, 230, 508
size, 13
structures, 115
sysadmsh(ADM) options, 36, 42, 96-97
troubleshooting, 554, 564-575
types, 99
unmount, 565
writing buffers to disk, 530
Filter, printer, 155, 159
find(C), 105-108, 566
Finding files, 105-106
ERROR: UNCLOSED RANGE ON PREVIOUS ENTRY
FIPS 151-1 conformance, NIST, 92
fixmog(ADM), 62, 232, 488
fixperm command, checking UUCP file permissions, 613
FLCKREC parameter, 534
Floating-point emulator error, 554
Floating-Point Unit (FPU), 55
Floppy, sysadmsh(ADM) selection, 186
Floppy disk, 181, 183
See also Disk, 181
blocks, 647
booting, 553
copying, 182-183
disadvantages, 455
filesystem, 184-186
formatting, 181
gap and block numbers, 648
security, 202
sysadmsh(ADM) options, 41, 184
Floppy drive, device names, 643
Fonts, 128-131, 637-638
Form printer, 146, 155
Format
mail, 377
mailbox, 437
Formatting
floppy disks, 181
floppy disks, sysadmsh selection, 41
FP reference section, 8
Fragmentation, 103, 111
Free list, 119
Free space, 102-103, 113
Free-block list, 119
fsave(ADM), 455
fsck(ADM), 114, 117-118
AFS filesystem, 99
boot option, 44
checking hard disk problems, 568, 571
DOS, use with, 198
EAFS filesystem, 99
failed mount command, 574
gap and block numbers, 648
mounting filesystems, 216
phases, 118
restoring root filesystem, 568
single-user mode, 486
system crash, 230, 508
troubleshooting considerations, 481
wrong ISIZE and FSIZE values, 572
fsdb(ADM)
correcting super block values, 573
disabling error checking, 574
DOS filesystems, 574
filesystem debugger, 572
repairing DOS filesystem, 118
repairing UNIX filesystem, 118, 574
FSIZE value, 572-573
fsphoto(ADM), 455, 477
"fsstat: root filesystem needs checking", 44

G

Gap number, 648
"Garbage or loose cable ... port shut down", 583
Gateway, MMDF, 418
getty
-c option, 292
default terminal line characteristics, 289
gettydefs file. *See* /etc/gettydefs file
GID. *See* Group ID
goodpw(ADM), 89
GOV domain name, 386
GPGSHI parameter, 537
GPGSLO parameter, 537
GPGSMSK parameter, 537
Graphics parameters, sysadmsh(ADM) selection, 37
Green Book, 209
gr_id_map file, 50
Group, 76-77

Group ID (GID), 203
  filesystem, 100
  setting directory SGID bit, 100
  supplemental groups, 77
Guest, UUCP transaction, 329

H

halt sys(ADM), 53, 54
Hard disk
  See also Disk
  adding, 36
  blocks, 647
  device names, 643
  gap and block numbers, 648
  heads, 648
  nonstandard disks, 200
  partitions, 188-190, 192, 194
  security, 202
  tracks, 189
  troubleshooting, 555, 568
Hardware devices
  displaying, 54-55
  errors, 482
Hardware failure
  defective printer, 587, 590
  modem port problems, 580
  non-functional terminal, 602
  super block damage, 571
  system panic, 486
  tape controller card, 600-601
  terminal, hung, 602-603
Hardware problems, 645
Hardware-dependent parameters, 546
Hardwired devices, 648
Hash queues
  increasing with system buffers, 528
  S5HASHQS parameter, 535
Hayes (and compatible) modem
  configuring, 308
  connecting, 330
  switch settings, 306-307
  testing, 333
hd(C), 112
Header files directory, 645
Heads, hard disk, 648
Hierarchy, sysadmsh(ADM), 35
Highlighting, sysadmsh, 20
Home directory
  changing, 70
  sharing, 66
  sysadmsh(ADM) selection, 36
  user, 57
homepaths file, 78
Host
  adapter number, SCSI tape drive, 600
  UUCP transaction, 329
Host name
  changing, 426
  hiding, 388
  underscore character, 385
HP printers, 124
"hsiread - extent table overflow", 494
"hsiread - inode table overflow", 494
HSNEXTENT parameter, 535
HSNINODE parameter, 535
HSNMOUNT parameter, 535
Hung process, 13
HUPCL, 290, 291
HW reference section, 8
hwconfig(C), 55

I

IBM Proprinter XL, 124
"id - Out of STREAMS", 494
"id - out of STREAMS queues", 542
ID number, SCSI tape drive, 600
id(C), 213
Identification and Authentication (I&A), 204
Identity, sysadmsh(ADM) selection, 69-70
Idle time, sysadmsh(ADM) selection, 36
Idle users, 211
idleout(ADM), 211
IDLETIME variable, 211
idmkenv(ADM), 556, 593
idtune(ADM), kernel parameters, 509
"ifreeget - Inode table overflow", 494, 532
Incremental backups, 474
inetd, 355
"INIT: execlp of /bin/sulogin failed", 557
initdefault action, /etc/inittab, 621
Initialization files, 625
  /etc/cshrc, 626
  /etc/inittab, 620
  /etc/motd, 627
  /etc/profile, 626

Initialization files, 625 *(continued)*
/etc/rc2.d scripts, 623
.login, 626-627
modifying, 619
.profile, 626-627
init(M)
boot problems, 556, 558
maintaining filesystems, 116
power failure signal, 622
run levels, 620
S command, 585
"INITSH: /etc/bcheckrc not found", 557
inittab file, 310
Inode table
allocating entries, 532
status, 520
Inodes, 115
See also Inode table
allocation, 572
available, 573
clearing, 119
count error, 114
directory, 112
filesystem problems, 564
null numbers, 112
parameters, 532
reconfiguring with mkfs(ADM), 565
shadow, 112
structure, 115
Installing
Emergency Boot Floppy Set, 552
fonts, 128-131
modem, 305-306, 308
planning system, 14
printer, 125-131, 171
integrity(ADM), 50, 62, 216, 229, 231-232, 509
Interface, printer, 123, 163-169
Internal modem, 309
Internet, 375
technical bulletin RFC822, 423
top-level domain codes, 386
Interprocess communication, sar -m, 516
Interrupt key, 9
Interrupt vector
boot display, 54
COM ports, 312
conflicts, 308
printer port conflict, 588
tape drive conflict, 599, 601
Invoking programs, users, 13
I/O, tuning, 494
IRQ (physical interrupt). *See* Interrupt vector
Irwin, tape drive, 180
ISIZE value, 572-573
ISO (International Standards Organization), 385
IXANY, 290

J

Job scheduling, 629-632
Job. *See* Process
Jumper settings, tape drives, 600

K

K files, 624
K reference section, 8
kbmode(ADM), 634
KDBSYMSIZE parameter, 532
Kernel
audit records, 239
authorizations, 90-93, 204, 208-209, 562
authorizations, chmodsugid, 629
environment, 556
managing virtual address space, 536
parameters, 235
KBTYPE, 634
relinking, 620
relinking with link_unix, 504
resources, 504, 509
sysadmsh(ADM) selections, 36
"Kernel: i/o bufs", 558
Kernel parameters, 525, 527
See also Parameters; System
asynchronous I/O, 548
boot load extension, 548
buffers, 528
changing, 504
clock, 538
devices, 545
disks, 528
ETRUNC, 100
event queues, 545
files, 532
filesystems, 532
idtune(ADM), 509
inodes, 532
memory management, 535

Kernel parameters, 525, 527 *(continued)*
message queues, 539
multiprocessing, 549
multiscreens, 538
overriding limits, 505
processes, 535
semaphores, 540
shared data, 541
streams data, 542
swapping, 535
sysadmsh(ADM), 36
system name, 542
table overflow, 493-494
Kernel problems
backing up, 560
booting from alternate kernel files, 553
inconsistencies causing system panic, 486
keyboard lockup patch, 560
out of inodes on EAFS /dev, 564
Keyboard
AT mode support, 634
bell, sounding, 639
changing default mode, 634
end input, 9
lockup, 559-560, 594
selecting types, 634
switching modes, 634
test with modem, 312
type, console, 560
UNIX keystrokes, 9
Keystrokes, 9
Kill key, 9
kill(C), 13
Korn shell (ksh)
.profile file, 296
TERM assignment, 295

L

Label field, /etc/inittab, 620
Labeling in backups, 461
LAN (Local Area Network)
MMDF, 417
planning, 14
UUCP, 338, 348-349
Landscape mode, 124
LCK (lock) files, 329, 334, 367-368, 370, 374
Leased line, 332
Level 0 backup, 459, 463
/lib directory, 644
Lineprinter, locating, 14
link_unix(ADM), 504
list channel, 413, 423
listen, 359
Listing, directory, sysadmsh selection, 41
ln(C), -s option, 101
Load management, printer, 147
Loading, operating system, 43
Local
channel, 423
domain, 386
network switch, 351-352
printing, lprint(C), 138
systems, connecting with direct wire, 360
Locale variables, sysadmsh(ADM) selection, 36
Locating files, 105-106, 108
Lock files, UUCP, 612
Locked
See also Keyboard
account, 576
device, 611
keyboard, 559-560
terminal, 604
Locking
account, 68, 596
login, 210
terminal, 57
Log book, 2-3, 462
.Log directory, UUCP programs, 610
Log files
See also Reports
cleaning, 108-110
clearing system files, 566
error messages, 645
/etc/ddate file, 108
monitoring, 110
password, 221
printer, 156-157
system, 646
/usr/adm/sulog file, 109
UUCP, 326, 367, 610
log(HW), 482
Login
activity reporting, 224
auditing, 283
configuring, 65-66, 85-86
creating dial-in accounts, 332
directory size, 112

Login *(continued)*
ending session, 9
lock out, 210
multiscreen, 11
procedure, 12
prompt, 52
reports, 39, 220
script (chat script), 335, 338-339
security, 38, 81, 83, 85-86, 211, 219-220, 354
sequence, UUCP, 327, 329, 339
shell, adding, 94
system boot, 45
troubleshooting, 555, 575-577, 602, 607
User ID (LUID), 547
.login file, setting system variables, 627
"Login incorrect", 598
"login: resource Authorization name file could not be allocated ...", 598
login(M), restoring, 555
Logins, sysadmsh(ADM) selection, 68, 85-86
LOGNAME entry, Permissions file, 341
Long filenames, 99
Loop, process in, 586
lost+found directory, 97, 119
lp authorization, 91, 209
lp* commands, 138
LPDEST, shell variable, 145
lpmove(ADM), 149-150
lprint(C), 138
lpsched(ADM)
enabling and disabling printers, 143
manually starting the print service, 142
print problems, 587
starting, 587, 589
lpshut(ADM), daemon, 142
lpstat(C), 136, 154-155
LPTELL shell variable, 168
LQP-40 printer, 124
LUID (Login User ID), 78, 203, 235, 238, 626
LUN (Logical Unit Number), SCSI, 600

M

M reference section, 8
MACHINE entry
OTHER option, 347
Permissions file, 341
Machine name
See also Host name
backup schedule, 456
Mail, 375
See also MMDF
administering, 435
etiquette, 13
header, 377
mail program, 105
routing, 378, 415
sending to remote sites, 323
troubleshooting, 577-578
/usr/spool directory, 645
UUCP, 327, 334
Mail Transport Agent (MTA), 375
Mail User Agent (MUA), 375
Mailbox, file, missing, 578
Maintenance, UNIX, 1, 2
MakeDir DOS, sysadmsh(ADM) selection, 42
man(C), 7
Manual pages, 7
mapflush daemon, 536
mapkey(ADM), 11
mapstr(M), 302
Mask, auditing, 251, 288
Master tty, 299
Masterboot block, nonstandard disk parameters, 200
MAXACPUS parameter, 549
MAX_BDEV parameter, 549
MAXBUF parameter, 529
MAX_CDEV parameter, 549
MAX_CFGSIZE parameter, 531
MAXFC parameter, 537
Maximum screens and pseudo-ttys, 299
MAXPMEM parameter, 536
MAXSC parameter, 537
MAXSEPGCNT parameter, 544
MAXSLICE parameter, 538
MAXUMEM parameter, 537
MAXUP parameter, 536
MCHANLOG, 450
MDLVRDIR, 437
Media
administering, 14
security, 202
sysadmsh(ADM) options, 41
types, 175
mem authorization, 91

Memory
image, 482
management, 535
shared segment size, 541
Menu, sysadmsh(ADM), 19-20, 35
menumerge(ADM), 35
mesg(C), 130
Message
queues, 539
system-wide, 52
UUCP, 607
Message of the day file. *See* /etc/motd file
messages file, 482
MFAILTIME, 449
MIL domain name, 386
MINARMEM parameter, 537
MINASMEM parameter, 537
MINHIDUSTK parameter, 537
Mini-cartridge, sizes, 178
MINUSTKGAP parameter, 537
mkdev(ADM)
fs, 96-97
ptty, 299, 539
tape, 176
mkfs(ADM), 647-648
backups, 565
reconfiguring inodes, 565
MLCKTYPE, 449
MLDOMAIN, 407
MLNAME, 407
MLOCMACHINE, 407
MMAXHOPS, 450
MMBXNAME, 437
MMBXPROT, 437
MMDF
See also Mail
administering, 435
configuring, 380, 388, 404, 407, 415, 417-418, 437
delivery, 380
hidden machine, 388
host name, 426
keywords, 407, 437
LAN (Local Area Network), 417
mail format, 377
mailbox file, 437
maintaining, 435
parameters, 437, 447, 449-450
problems, 577
queues, 435
MMDF *(continued)*
routing, 377-378, 415, 424, 437
testing addresses, 435
testing configuration, 434
UUCP, 334
UUXSTR, 431
XENIX file conversion, 428
MMDF alias
See also Alias file
ALIAS entry, 447
creating file, 446
file, 412
search sequence, 415
table, 446-447
MMDF channel, 422
badhosts, 380, 423
badusers, 380
configuring, 418
list channel, 413
types, 423
mmdfalias(ADM), 429
mmdftailor(F), 404, 578
MMSGLOG, 450
MNP4, 583
/mnt directory, 184, 644
mnt(C), 98
Mode
changing, 9
single user, 45-46
system maintenance, 45-46
Modem, 305
auto-answer, 313
baud rate, 330
checking baud rate, 580
checking /etc/gettydefs file, 584
checking modem cable, 579
configuring, 306-308
connecting, 330
data compression, 583
dialers, 315
Dialers file, 329, 351-353
dial-in, 313-314
dialing configuration, 307
dial-out, 313, 315
editing /etc/inittab file, 310
error correction, 583
Hayes (and compatible), 306-307, 330, 333
incompatible, 583
installing, 305-306, 308
local network switch, 352

Modem, 305 *(continued)*
login sequence, 339
null modem, 309
pin connections, cabling, 309
planning, 14
printer, 171
printer connection, 127
problems in UUCP, 611
speed conversion, 319
supported, 325
switch settings, 307
telephone line, 323
testing, 311, 333
testing phone line, 580, 611
Trailblazer, 308
troubleshooting, 579-584
UUCP use, 309
volume, 308
MODE_SELECT parameter, 546, 593
Mount table, 533
mount(ADM), 216, 647
boot floppy, 553
failed, 574
hard disk root filesystem, 555
use of, 97, 186
Mounting, 97
directory, 644
filesystem, 96-98, 216
floppy disk directories, 184-186
print wheel, 130-131, 145-146
Mouse
installing, 37
troubleshooting, 585
/usr/lib/event/ttys, 585
Moving, print jobs, 149-150
"MPX timeout - Timeout table overflow", 494
"MPX timeout - Timeout table overflow", 538
mscreen(M)
configuring, 298
example, 301
mscreencap file, 299
troubleshooting, 299
tuning, 300
MSGMAP parameter, 539
MSGMAX parameter, 539
MSGMNB parameter, 539
MSGMNI parameter, 539
MSGSEG parameter, 540
MSGSSZ parameter, 539
MSGTQL parameter, 539
MSLEEP, 450
MSUPPORT, 407
MTA (Mail Transport Agent), 375
mtune(F), parameters file, 526
MUA (Mail User Agent), 375
Multiplexer links, 543
Multiport serial card, 306
Multiprocessing, kernel parameters, 549
MultiScreen, 10
access, 635
console, 635
diagnosing keyboard lockup, 559
example, 10
multiple video adapters, 637
parameters, 538
troubleshooting, 597
multiscreen(M), 298
Multiuser mode, 45
cannot enter, 556
no login on console, 561
system hangs at login prompt, 555
MWARNTIME, 450

N

NAHACCB parameter, 546
NAIOBUF parameter, 548
NAIOHBUF parameter, 548
NAIOLOCKTBL parameter, 548
NAIOPROC parameter, 548
NAIOREQ parameter, 548
NAIOREQPP parameter, 548
Name cache, 517
Naming files, sysadmsh(ADM) selection, 42
NAUTOUP parameter, 529
NBLK parameter, 542
NBLK<n> parameter, 543
NBUF parameter, 528
NCALL parameter, 538
ncheck(ADM), 217
NCLIST parameter, 531
NCOPYBUF parameter, 529
NCPYRIGHT parameter, 531
NDISK parameter, 528
NEMAP parameter, 531
NET domain name, 386
Network
planning, 14
printer, 170

Network *(continued)*
sysadmsh(ADM) selections, 36
UUCP, 324
Network address, for UUCP over TCP/IP, 357
Network Listing Service (NLS), 358
newgrp(C), 77
"newproc - Process table overflow", 494
NFILE parameter, 533
NGROUPS parameter, 77, 535
NHBUF parameter, 528
NHINODE parameter, 532
NIC (Network Information Center), 387
nice(C), 632
NINODE parameter, 532
ERROR: UNCLOSED RANGE ON PREVIOUS ENTRY
NIST FIPS 151-1 conformance, 92
NKDVTTY parameter, 531
NLOG parameter, 545
nlsadmin(ADM), with UUCP, 358
NMOUNT parameter, 533
NMPBUF parameter, 534
NMPHEADBUF parameter, 534
NMUXLINK parameter, 543
"NO DEVICES AVAILABLE", 607
"No more processes", 586
No outgoing calls, UUCP messages, 613
Node, device, 642
NODE parameter, 542
NOFILES parameter, 533
None option
at and batch program defaults, 630
cron program defaults, 630
NOREAD option, Permissions file, 344
Normal operation, 45, 53
"NOTICE: clalloc - Out of space on EAFS dev hd", 565
"NOTICE: s5ialloc - Out of inodes on EAFS /dev", 564
NOWRITE option, Permissions file, 344
NPBUF parameter, 528
NPROC parameter, 535
NQUEUE parameter, 542
NREGION parameter, 536
NSCRN parameter, 539
NSHINTR parameter, 546
NSPTTYS parameter, 299, 539
NSTREAM parameter, 542, 543
NSTREVENT parameter, 543
NSTRPUSH parameter, 543
Null modem, 309
NUMSP parameter, 545
NUMSXT parameter, 531
NUMTIM parameter, 545
NUMTRW parameter, 545
NUMXT parameter, 531
nuucp account, 356

O

Objects, 203, 237
off action, /etc/inittab file, 622
OK message, cu(C), 311, 579
once action, /etc/inittab file, 622
ondemand action, /etc/inittab file, 622
open(S), 242
Operating system
See also System
loading, 43
reinstalling, 552, 568
Operation modes, 45
Orange Book, 12, 201
ORG domain name, 386
OS/2, 188
OTHER option, MACHINE entry, 347
"Out of clists", 494
Out of clists (NCLIST = <number> exceeded), 531
Override terminal, 230, 595

P

pacct log file, 646
Packet, UUCP, 329, 354
Page and swap usage, sar -r, 519
Page size, printer, 135
Paging activity, sar -p, 517
Panic reboot, 56
Parallel printer, 172
Parameters
See also Kernel parameters
audit, 37, 257
boot, 55-56
login, 86
MMDF, 407, 437, 447, 449-450
UUCP, 354
Parity values, modem problems, 581, 584
Partition, table, 189

Partitioning hard disk
  assigning, 190
  DOS, 193
  fdisk(ADM), 189
  installing UNIX on DOS machines, 192
  removing, 194
  two hard disks, 193
passwd file. *See* /etc/passwd file
Password
  activity reporting, 221
  assigning, 62-63, 72, 88
  auditing, 284
  C2 security, 205
  changing, 57, 71-72, 576
  checking, 89
  compatibility with other UNIX systems, 79
  dial-in, 314
  disabling checks, 89
  expect string, 338-339
  expiration, 39, 72-73, 87-89, 209-210
  Management Guideline, 209
  matching and rejecting, 90
  reports, 220, 222
  restrictions, 209
  root, 52
  security, 87, 226, 228
  super user, 9
  sysadmsh(ADM) options, 38-39
  Systems file, 327, 329
  user generating, 12
  UUCP logins, 333
PATH variable, checking, 490
Pathname, {} command argument, 106
Performance, tools, 503
Permission
  archiving, 216
  directories, 212
  file, 212
Permissions
  locating files, 106-107
  mailbox file, 437
  problems,sticky bit setting, 575
  serial port, 309, 361
  sysadmsh(ADM) options, 41
  viewing user authorizations, 631
Permissions file, 327, 341
  combining entries, 347
  configuring, 342, 347
  errors, 371, 373
  example, 329, 332
Permissions file, 327, 341 *(continued)*
  file transfer, 328
  format, 341, 347
  granting access, 346-347
  options, 342
Phone field, Systems file, 337
Phone line
  checking for noise, 581
  modem problems, 580, 611
Physical security, 202
pica pitch, 135, 165
PIOMAP parameter, 530
PIOMAXSZ parameter, 530
Pitch, printer, 135
Planning, installation, 14
PLOWBUFS parameter, 529
PN and PS termcap entries, 138
Polling, setting for UUCP, 340
Populate existing, sysadmsh(ADM), 66
Port
  printer, 159-161
  problems, 588-590
  serial, 305, 309
Portrait mode, 124
Ports, settings changed with stty(C), 293
postmaster, routing, 415
PostScript, options, 124
PostScript printer, 124
Post-selection, auditing, 288
Power cycle, 13
Power failure, recovering from, 485
Power supply, 14
powerfail action, /etc/inittab file, 622
Powering on, 43
powerwait action, /etc/inittab file, 622
Preprinted forms, 40
Preselection, 288
PRFMAX parameter, 531
Primary authorizations, 92
PRINT port, 138
Print service, restricting user access, 134
Print wheel, 40, 128-131, 146
Printer
  accepting, 39
  adding, 37, 39, 138
  administering, 134
  cabling, 138
  class, 144-145, 147
  configuring, 39, 122-124, 154-155, 172
  connection port, 127

Printer *(continued)*
defaults, 39, 135-136, 145
dialup, 171
disabling, 143
enabling, 143
fault, 131-134
filters, 159
fonts, 128-131
forms, 145
initializing, 172-173
installing, 14, 125-131, 171
interface program, 123, 163-169
local printing: lprint(C), 138
log files, 156-157
modem connection, 127
modes, 124
moving jobs, 149-150
network, 170
parallel, 172
rejecting, 39
removing, 37, 39
starting, 150
status, 36
terminfo, 161-163
terms, 122
troubleshooting, 587-594
types, 125-127
Printer port
adjusting, 159
characteristics, 160-161
initializing, 160
printerstat authorization, 92, 137
Printing
administering, 121-122, 147-157
cancelling request, 143
command summary, 136-137
configuring, 157-171
daemons, 158
defaults, 152-154
explaining to users, 13
filters, 40
forms, 40
load management, 147
moving jobs, 149
priority, 150
queue, 40, 150-154
requests, 39, 121, 143
retry rate, 171
scheduler, 142
spooling, 121, 158-159, 172
Printing *(continued)*
starting, 142, 148
stopping, 142, 148
sysadmsh(ADM) options, 39, 42
printqueue authorization, 92
Printronix printer, 124
Priority, print job, 150
Privileges. *See* Authorization
Privileges, sysadmsh(ADM) selection, 66, 74-75
Process
in loop, 586
memory management and swapping, 535
parameters, 535
regions, 536
runaway, 13
scheduling, 41
security, 203-204
specifying maximum time slice, 538
sysadmsh(ADM) options, 41
troubleshooting, 586, 594-595
Process accounting, 524
Process field, /etc/inittab file, 620
Process table, allocating entries, 535
.profile file, 295-296, 627
Program
simultaneous execution, 10
system specific, 38
Prompt, login, 52
Protected password database, 58, 80, 228
Protected subsystems, 205-206
Protocol, 375
prwarn(C), 74, 88, 210, 222
ps(C), 13
checking process activity, 510
Pseudo-ttys
default, 539
master and slave, 643
pstat(C), 521, 524
ptys, 298
Public, directories, 215
putbuf buffer, 530
PUTBUFSZ parameter, 530
pwconv(ADM), 58
pw_id_map file, 50

Q

QIC-02 tape drive size, 177
queryspace authorization, 92
Queues
  MMDF, 435
  printer, 40, 121, 150-154
Quitting
  login session, 9
  sysadmsh(ADM), 42
Qume Sprint 1155 printer, 124
quot(C), 104-105

R

rc directories, 643
"Reached end of medium on output", 464, 467
READ option, Permissions file, 343
Reboot. *See* Boot
Reboot system, 38
Records, locked by system, 534
Recursion, 106-107
Region table, 536
reject(ADM), 148
Rejecting, printer requests, 148
Relative pathnames, 179
Relaxed security, 36, 83, 227
Remote
  command execution, 323, 330
  computer link, 328
  UUCP connection, 330
remote.unknown script, 354
Removing
  printer class, 147
  user, 57, 67
Report template, audit, 267, 274-276
Reports
  security, 211
  sysadmsh(ADM) options, 220-224
Request ID, printing, 121, 143
Request log, printer, 156
REQUEST option, Permissions file, 342
Reset, audit rules, 37
Resetting terminal, 13
Resources. *See* System resources
respawn action, /etc/inittab file, 622
Restart system, 38
Restoring
  filesystems, 487, 565, 568
  space, 566
  system files, 552-558
Retire, sysadmsh(ADM) selection, 67
Retry period, UUCP, 336
Return key, 9
Reverse video, colors, 639
rmail(ADM), 342, 345, 347
rmuser(ADM), 67
Root
  See also Super user
  backups, 454, 476, 478
  boot, 45
  directory, 641
  login, 52
  password, 52
  security, 65, 230, 595
root authorization, 77
Root filesystem, 95
  mount table, 533
  restoring, 568
Root floppy
  Emergency Boot Floppy Set, 552, 558
  restoring /etc/emulator file, 554
  restoring /etc/init file, 558
  restoring root filesystem, 568
Routing
  files,converting, 428
  MMDF, 334
RS-232
  connecting the cable, 361
  interface, 121
  selecting a serial port, 360
  UUCP requirement, 325
Run level
  changing with telinit(M), 620
  field, /etc/inittab file, 620
  system default, 622
Runaway process, 586
  killing, 594

S

S files, /etc/rc2.d directory, 624
S reference section, 8
S00SYSINIT script, 624
S01MOUNTFSYS script, 624
S03RECOVERY script, 624

S04CLEAN script, 624
S05RMTMPFILES script, 624
S15HWDDNLOAD script, 624
S16KERNINIT script, 624
S20sysetup script, 624
S21perf script, 624
S5CACHEENTS parameter, 535
S5HASHQS parameter, 535
"s5iread - ... inode table overflow", 494
S5OFBIAS parameter, 535
S70uucp script, 624
S75cron script, 624
S80lp script, 624
S86mmdf file, 432
S86mmdf script, 624
S87USRDAEMON script, 624
S88USRDEFINE script, 624
S90RESERVED script, 624
sadc (System Activity Data Collector) command, 512
SA_EDITOR, 456
Safe to Power Off message, 53-54
Safety, 14
SANE, 290-291
sar(ADM), 511-522
Scan window, 30, 35
scancode terminal, 302
scanoff(M), 303
scanon(M), 303
Schedule field, Systems file, 335
schedule file, 458
schedule(ADM), 455
  file, 456
Scheduled backups, 456-465
Scheduler
  printing, 142
  problems, 561, 587
"scheduler is not running", 587
Screen saver, TBLNK parameter, 538
Screens
  maximum, 299
  multiple, 10
  protection
    console, 635
    TBLNK parameter, 635
Scripts, adding to init procedure, 625
SCRNMEM parameter, 539
SCSI, bad block handling, 563
scsibadblk, 563
sd(ADM), 233, 626
sdd daemon, 233, 626
SDSKOUT parameter, 546
Search. *See* find(C)
SECCLEARID parameter, 235, 547
SECLUID parameter, 235
SECLUID parameter, 547
Secondary authorizations, 92
SECSTOPIO parameter, 236, 547
Sector, track, 648
Security
  abuse of privilege, 227
  accountability, 203
  auditing, 205, 212
  authorizations, 77, 90, 204, 208, 211
  C2, 12, 201-210
  crash recovery, 229, 487
  daemons, 233-234
  database, 47, 58-61, 228-231, 487, 595
  defaults, 81, 83
  disabling C2 features, 235, 547
  encryption, 215
  failure mode, 83
  features, 48-50, 202-210, 212-213, 215, 225, 227-231, 488, 508, 595
  filesystem, 212, 227
  high, 83
  Identification and Authentication (I&A), 204
  importing data, 215
  improved, 83
  kernel parameters, 547
  level, informing users, 12
  login, 211
  low, 83, 93
  maintained, 201
  parameters, 81, 83
  password, 87, 209, 226
  physical, 202, 227
  problems, 232, 509
  reports, 39, 220-224
  retirement, 83
  sticky bit, 213
  subsystems, 205, 207
  super user, 9
  sysadmsh(ADM) options, 36, 38-39, 207
  system access, 209
  tampering, 225
  tcb directory, 645
  terminal, 217, 219-220, 229
  terminal restrictions, 210

Security *(continued)*
traditional, 93
troubleshooting, 595-599
UID reuse, 83
UNIX system, 83
UUCP, 345-346
SEMAEM parameter, 541
Semaphores, 540-541
SEMMAP parameter, 540
SEMMNI parameter, 540
SEMMNS parameter, 541
SEMMNU parameter, 540
SEMMSL parameter, 540
SEMOPM parameter, 540
SEMUME parameter, 540
SEMVMX parameter, 541
Send string, login script, 338
SENDFILES option, Permissions file, 342
Sending, message, 52
Serial
console, 297
multiscreens (mscreen), 298
Serial cable, 361
Serial card
multiport, 306
sysadmsh(ADM) selection, 37
Serial line
conflicts, 314
connecting, 14
display characteristics, 293
filenames, 648
lock file, 329
settings, standard, 584
UUCP requirement, 325
Serial port
choosing, 305
permissions, 309, 361
UUCP connection, 360-361
serial(HW), 309
set init example, 301
set noglob example, 301
set prompt example, 301
setcolor(C), 638-639
setenv TERM, 295
setgid(S), system call, 205, 244
setkey(C), 302
Setting, terminal type, 294, 296
setuid(S), system call, 205, 244
sg(C), 77
SGID bits, 208, 212-213, 216, 235, 547
sh, TERM assignment, 295
Shadow inodes, 112
Shared
data parameters, 541
libraries, specifying maximum number, 534
Shell
escape, sysadmsh(ADM), 34
exiting, 9
layers, 36
login, 65
SHLBMAX parameter, 534
shl(C), 299
SHMALL parameter, 541
SHMMAX parameter, 541
SHMMIN parameter, 541
SHMMNI parameter, 541
SHMSEG parameter, 541
Shutdown
abnormal, 485, 571
menu option, 38
procedure, 485
shutdown(ADM)
authorization, 92
notification, 52
procedure, 53-54
single-user option, 117
sysadmsh utility, 46
system maintenance mode, 46
SIGHUP signal, 168
SIGINT signal, 168
SIGPIPE signal, 168
SIGQUIT signal, 168
simple content type, 126
Single-user mode, 45
cannot enter, 557
changing kernel parameters, 504
fixing a bad track, 563
restoring /unix, 554
shutdown(ADM), 117
sio file, 310
Site planning, 14
Sitename
backup schedule, 456
verifying unique, 613
Size, displaying, 104, 105
Slash (/), root filesystem, 95
Slave tty, 299
slot(ADM), 55

Smartmodem
  1200 or compatible, 306
  2400, V-series 9600, 307
smmck script, 48
SMTP, MMDF, 423
"sp: spclose - Cannot allocate STREAMS block", 494, 543
Special
  device file, 361
  device node, creating, 592
Special filenames, 647
Speed conversion, 319
Speed field, Systems file, 337
Spooler
  printer, 121, 172
  UUCP, 324
SPTMAP parameter, 536
SRI International, 387
Stack region, 536
"Stage 1 boot failure: error loading /boot", 552
Starting
  print service, 142, 148
  printer, 150
  system, 43
  terminal output, 9
  UUCP network, 324-325
Startup, mounting filesystems, 97
Statistics, audit, 37
STATUS error messages, 372
.Status file, error, 372
Status files, UUCP, 610, 612
Status Line (sysadmsh), 19
Status report access, 211
Sticky bit, 213, 215, 234
Sticky bit setting, unable to remove files, 575
stopio(S), 236, 547
Stopping
  login session, 9
  print request, 143, 148
  print service, 142
  program, 9-10
  sysadmsh(ADM), 42
  system, 38, 52-54
  terminal output, 9
Storing files, 41, 175, 178, 181, 183
Straight-through cables, 309
STRCTLSZ parameter, 544
STREAM-head structures, 543
STREAMS, 542
  parameters, 542
  sysadmsh(ADM) option, 36
  viewing crash(ADM) usage statistics, 506
"string: Configuration buffer full", 494, 531
STRLOFRAC parameter, 544
STRMEDFRAC parameter, 544
STRMSGSZ parameter, 544
"stropen1 - Out of streams", 494, 543
"stropen2 - Out of streams", 494
stty(C), 293
  adjusting printer port characteristics, 160
  default options, 160
  printer port characteristics, 160-161, 166
stune(F), 527
Subdomain, 386
Subject files, 288
Subjects, 203
submit(ADM), 379
Subsystem, 205
  administrative roles, 206
  audit, 93, 205, 250, 284-285
  authorizations, 91-92, 204, 207, 209
  database boot message, 47
  database files, 229
  kernel, 209
  manual page, 208
  protected, 205
  sysadmsh(ADM) options, 207
su(C), 233, 238
  accessing other accounts, 78
  authorization, 78
  security, 65
SUID bits, 208, 212-213, 216, 235, 547
sulog file, 110
sulog log file, 646
Super block, 116
  correcting values, 573
  repairing damage with fsdb(ADM), 571
Super user, 8
  See also Root
  exiting, 52
  login, 52
  security, 9, 52, 65
  single-user mode, 45-46
.suppgroups file, 77
suspendaudit authorization, 92, 242, 288
"swapdel - Total swap area too small", 494, 537

Swapping
memory, 535
parameters, 535
reporting activity, 521
Switch, settings, modem, 307
Switching
activity, reporting, 521
operating systems, 190-191
screens, 10
Symbol table, size, 532
Symbolic links, 99, 101
sync(ADM), 116-117
sysadmin authorization, 91, 209
sysadmsh(ADM), 6, 17
backups, 454-456, 463-474, 565
color selection, 35
customizing menus, 35, 36
error messages, 20
files, restoring, 471
filesystem options, 96-97
keystrokes, 24-27
menu, 6, 19-28, 35
mount filesystem, 565
options, 19-22, 36-37
printer options, 123, 125, 127-128
Privileges options, 209
quitting, 42
restoring filesystem, 565
shell escape, 34
shortcuts, 35
subsystem options, 207
system shutdown, 46
unmount filesystem, 565
user selection, 21
Sysfiles file, 353
sysinit action, /etc/inittab file, 621
System
See also Console; Kernel, parameters
See also Kernel parameters; Operating system
buffer cache, tuning, 494
call activity, sar(ADM) -c, 514
checking status of tables, 520
cleaning, 44
crash, 229, 250, 487
customizing startup, 619
defaults, 36
defaults database, 59
diagnosing inefficiency, 503
error log, 645
System *(continued)*
files, 552-558, 566
hardware, 54-55
increasing reliability, 530
initialization files. *See* Initialization files
log, 482
maintenance, 1-5, 8, 102, 108-110
maintenance mode, 45-46
name, 426, 542
panic, 486-487
planning, 14
reports, 220-224
run level, 620, 622
safety, 14
security, 48-50, 202-210, 212-213, 215, 225, 227-231, 488, 508, 595
setting variables, 627
shutdown, 38, 52-54
shutdown, abnormal, 485
starting, 43
stopping, 38, 46, 52
time, 46
System Activity Reporter. *See* sar(ADM)
System activity, sysadmsh(ADM), 36
System administrator, 1-2
log book, 2-3
super user login, 8
tasks, 3-5
tools, 6
training users, 12
trusted system, 206
user maintenance, 57
System audit mask, 288
"SYSTEM NOT IN Systems FILE", 606
System resources
buffers, 525
kernel, 504, 509
limits, 525
Systems file
baud rate, 330
device requirements, 328
error, 373-374
error with cu(C) command, 606
examples, 329, 331
for TCP/IP, 356
format, 335
Local Area Network, 338
login sequence, 327, 329
phone field, 337
remote computer link, 328

Systems file *(continued)*
schedule field, 335-336
security, 334
speed field, 330, 337
TCP/IP address example, 359
testing modem, 333
UUCP, 327, 611-612
verifying phone number, 334
System-wide message, 52
SYSTTY, 297

T

-t files, 48-49
TAB3
/etc/gettydefs, 291
example, 291
Table entries, 525
Table overflow, 493-494
Tabs, changing, 294
tail(C), 110
Tampering, 225
Tandy DMP printers, 124
Tape
archiving files, 178
buffers, values, 529
device names, 643
/etc/default files, 177
sysadmsh(ADM) options, 41
troubleshooting, 599-601
Tape drive, 175
DAT, 178
default device, 176
Error Correction Code (ECC), 180
/etc/default files, 177
Exabyte (8mm), 178
Irwin, 180
maintaining, 179
mini-cartridge, 178
performance, 529
QIC-02, 177
sizes, 177
sysadmsh(ADM) selection, 36
tape(C), 179
tar(C)
archiving files, 178, 184
cannot open /dev/rct0, 601
default settings, 177
directories, 178
tar(C) *(continued)*
pathname limit, 179
relative versus absolute pathnames, 179
security, 216
TBLNK parameter, 538, 635
tcb, boot message, 47
/tcb directory, 645
TCB (Trusted Computing Base), 203
/tcb/audittmp files, 109
tcbck(ADM), 47-50, 230-231, 488, 598
/tcb/files/auth directory, 58
TCP dialer type, 349
TCP socket interface, with UUCP, 355
TCP/IP, printer configuration, 170
TCSEC (Trusted Computer System Evaluation Criteria), 81, 201
Telebit modem, 308
telinit(M), 620
modifying /etc/inittab, 623
system run level, 620
Temporary
files, locating, 106
files, removing, 102, 108
TERM environment variable, 294
printing, 128, 161-162
terminal type, 603
termcap file, 138
Terminal
activity reporting, 223
administering, 38
authorization, 91
baud rate, 290-291
characteristics, displaying, 293
connecting local printer, 138
control entry, cannot rewrite, 576
disabled, 598
disabling, 296
display, 9
enabling, 36, 598
I/O, sar -y, 522
lines, setting, 289
locking, 69, 219
override, 230, 595
reports, 220
restricting access, 210
scancode, 302
security, 217, 219-220, 229
settings, checking, 292
sysadmsh(ADM) options, 36, 38, 217, 219-220

Terminal *(continued)*
troubleshooting, 601-604
type, setting, 294, 296
unlocking, 69, 219
Terminal Control database, 229
"Terminal is disabled -- see Account Administrator", 598, 604
"Terminal is disabled but root login is allowed", 596
terminfo(F), 127, 161-163
termio(M), 290
Testing, MMDF, 434-435
Texas Instruments 855 printer, 124
Text region, 536
"The security databases are corrupt", 595
"There may be a system dump memory image in the swap device", 482
Time
setting, 46
sysadmsh(ADM) selection, 36
used by command, 523
TIMEOUT, 56
"timeout - Timeout table overflow", 494, 538
timex(ADM), 523
TIMEZONE parameter, 538
TLI dialer type, 349
TLI (Transport Layer Interface), UUCP, 357
TLIS dialer type, 349
TM. (temporary data) files, 329, 367
/tmp directory, 102, 215, 645
Top-level domain, 385
Trailblazer modem, 308
Training, for new users, 12
Transparent mode, 140
Transport Library Interface (TLI), 545
Troubleshooting
MMDF, 435
utilities, UUCP, 615
Truncation, filename, 100
Trusted
See also Security
alias table, 447
applications, 243
Computing Base, 203-210, 229, 597, 645
system, 12, 88, 202, 206, 215, 227
tset command, reinitializing terminal, 603
tset(C), 13, 301-302
tty ports, setting with stty(C), 293
tty(C), 296
ttys
database, 598
device file, 648
modem, 306
root, 230, 595
ttyupd(ADM), 51
Tunable kernel parameters. *See* Kernel parameters
TZ (timezone) variable, 538

U

UID. *See* User ID
ULIMIT parameter, 536
umask(C), 212-213
umount(ADM)
floppy filesystem, 553-554
hard disk root filesystem, 555
use of, 97, 117, 186
Unattended backups, 477
Undo structures, number in system, 540
Unique, sitenames, UUCP, 613
UNIX
See also System
/dev/hd0a, device name, 189
filesystem, repairing with fsdb(ADM), 574
keyboard, 9
partition on DOS systems, 192
removing partition, 194
/unix, restoring, 553
/unix file, root directory, 642
"unix not found", 553
/unix.old, alternate kernel file, 55, 553
Unkillable process, removing, 594
Unknown, printer type, 125
"UNKNOWN FILE SYSTEM VERSION", 568
"Unknown Host Domain", 578
Unlocking
account, 596
console keyboard, 559-560
terminal, 57, 604
user account, 68, 576
Unmounting, print wheel, 146
Unscheduled backups, 454, 465-468
"usemouse: can't open slave (13)", 585
"usemouse: no mouse available(5)", 585
User
adding, 57, 60-63, 66
administrative, 66

User *(continued)*
auditing, 75
authorizations, 74-75
disk usage, 104-105
group, 69
home directory, 66
locking account, 68
modifying accounts, 64-76, 81, 83
printer access, restricting, 134
removing, 57, 67
reports, 220-222
retiring, 107
security, 225
training, 12
type, 65
unretiring, 67
User audit mask, 288
User ID
login field, 65
security, 203
"useshell: File Control database inconsistency", 62, 598
"useshell: Script path /usr/lib/mkuser/mkuser.init may be compromised ...", 598
/usr directory, 645
/usr/adm directory, 645
/usr/adm/messages file, 109
/usr/adm/pacct file, 109
/usr/adm/sulog file, 109
/usr/bin directory, 325, 645
/usr/include directory, 645
/usr/include/sys/netinet/in.h, 357
/usr/include/sys/socket.h, 357
/usr/lib directory, 645
/usr/lib/cron/.proto file, at(C) program, 632
/usr/lib/cron/.proto.b file, batch(C) program, 632
/usr/lib/event/ttys, 585
/usr/lib/goodpw/checks directory, 89
/usr/lib/goodpw/match file, 90
/usr/lib/goodpw/reject file, 90
/usr/lib/keyboard/strings.d, 303
/usr/lib/sysadmin/schedule, 456
/usr/lib/terminfo file, printer types, 591
/usr/lib/uucp, 326-327, 330
/usr/lib/uucp/uucico, login shell, 333
/usr/lib/vidi, directory, 637
/usr/mmdf/chans, 422
/usr/mmdf/log, 435
/usr/mmdf/mmdftailor, 404
/usr/mmdf/table directory, errors, 577
/usr/net/nls/inet/tcp directory, 358
/usr/spool directory, 645
/usr/spool/lp/logs/request files, 109
/usr/spool/lp/model file, 163
/usr/spool/mail, 435, 578
/usr/spool/mmdf, 435
/usr/spool/uucp, 326, 610
/usr/spool/uucp/.Log directory, 610
/usr/spool/uucp/LOGFILE file, 109
/usr/spool/uucp/.Log/.Old files, 109
/usr/spool/uucppublic, 326
/usr/spool/uucp/.Status directory, 610
/usr/tmp directory, 645
Utilities, accessing DOS, 194
uucheck(ADM), 326, 613, 615
uucico(ADM), 610
uucico(C), 324, 326-329, 354
uuclean(ADM), 326
UUCP, 323-324
access, 327
ACU (Automatic Call Unit), 337, 349
administering, 366
configuration files, 36
configuring, 330, 335-336
connecting, 324-325
creating login accounts, 332
daemons, 326-329, 345-346, 354
default paths, 345
error messages, 369-370, 372
examples, 362
granting access, 344-347
leased line, 332
linking, 349
Local Area Network (LAN), 348-349
login, 332-333, 335, 338-339
messages, 610-613
MMDF, 386, 423
modem, 309, 325, 330, 333
node name, 335
over TCP/IP, 349, 354
password, 333
polling, 340
port, 309
printer configuration, 170
protocols, 352
sample transaction, 328
security, 327, 334, 336, 341, 344-346, 354
setting up, 325

UUCP, 323-324 *(continued)*
status, sysadmsh(ADM) selection, 36
troubleshooting, 594, 604-615
UUCP commands
permissions problems, 607
remote execution, 330
uucheck(ADM), 615
uucico(ADM), 610
uulog(C), 608
uustat(C), 608
uutry(ADM), 609
UUCP directories
public directory, 342
security, 344
spool directory, 367
/usr/bin/uucp, 327
/usr/spool/uucp, 326
/usr/spool/uucppublic, 326
"uucp failed ...", 607
UUCP files
C. (work) files, 326, 329, 367-368, 370-371
control files, 324, 329-331
D. (data) files, 326, 367, 369, 371
database control files, 324
Devices file, 327, 330, 348, 349
Dialcodes file, 353
Dialers file, 315, 329, 350-353, 374
editing, 331
LCK (lock) files, 329, 334, 367-368, 370, 374
.Log directory, 610
log files, 109, 326, 329, 366-367, 646
Permissions file, 327-328, 342
permissions problems, 613
routing files, converting, 428
.Status directory, 610
Sysfiles file, 353
Systems file, 335-337
TM. (temporary data) files, 329, 367
uudemon.hour, 328
uulog(C), 367
X. (execute) files, 326-327, 330, 367, 369-370
uucp(C), 315, 323, 325-326, 328, 607-609
uucppublic directory, 215, 343
uudemon.admin script, 366
uudemon.clean script, 109, 326, 366-367
uudemon.hour script, 328, 366
uudemon.poll script, 366
uuinstall(ADM), 324
uulist(ADM), 429
uulog(C), 326, 367, 608, 615
UUname, 407
uuname(C), 366, 429, 615
uupick(C), 325
uusched(ADM), 327-329
uustat(C), 326, 608, 615
uuto(C), 326, 367
uutry(ADM), 333, 339, 374, 609, 615
uux(C), 323, 345
displaying status of transfer, 325
executing remote commands, 326
location, 327
problems, 615
records, 326, 367
specifying options, 431
UUCP channel relationship, 423
uuxqt(ADM), 345
executing remote programs, 327, 330
MACHINE option, 346
querying, 367
X. (execute) files, 327

V

v.42bis, 583
VALIDATE entry, Permissions file, 346
Variables
CRDELAY, 140
FORMS, 141
prototype file, 632
setting, 627
Ventilation, 14
VGA_PLASMA parameter, 546
vhand daemon, 536
Video adapters, multiple, 637
Video fonts, changing, 637
vidi utility, changing video fonts, 637
Viewing, file, 41
Virtual, terminals, 531, 539
vmstat(C), 515, 518, 521, 524
Volume, modem, 308

W

wait action, /etc/inittab file, 622
wall(ADM), 52, 103
"wd: ERROR on fixed disk ...", 563
Wheel, print, 128, 130-131
Window, scan window, 30
Wiring, modems, 309

WRITE option, Permissions file, 343
writeaudit authorization, 92, 208, 242, 288
write(C), 130
wtmp log file, 646
WYSE 60 termcap entry, 139
Wyse 60 terminal, 302

X

X. (execute) files, 326-327, 330, 367, 369-370
xargs(C), 107
xbackup(ADM), 455, 461
XENIX
  backups, 461
  filesystem, repairing with fsdb(ADM), 574
  MMDF, 428
  shared data segments, 541
  special semaphores, 541
XON/XOFF, printer, 131
xrestore(ADM), 455
"xsd_alloc - XENIX semaphore table overflow", 494
"xsd_alloc - XENIX shared data table overflow", 541
XSDSEGS parameter, 541
XSDSLOTS parameter, 541
"xsem_alloc - XENIX semaphore table overflow", 541
"xsemfork - File table overflow", 534
XSEMMAX parameter, 541
XT console keyboard, 560, 634
"xtinit - Cannot allocate xt link buffers", 494, 531

Y

"You do not have authorization to run ...", 599

Z

"Zero length FP emulator file", 554

Please help us to write computer manuals that meet your needs by completing this form. Please post the completed form to the Publications Manager nearest you: The Santa Cruz Operation, Ltd., Croxley Centre, Hatters Lane, Watford WD1 8YN, United Kingdom; The Santa Cruz Operation, Inc., 400 Encinal Street, P.O. Box 1900, Santa Cruz, California 95061, USA or SCO Canada, Inc., 130 Bloor Street West, 10th Floor, Toronto, Ontario, Canada M5S 1N5.

Volume title: _____
(Copy this from the title page of the manual)

Product: _____
(for example, SCO UNIX System V Release 3.2 Operating System Version 4.0)

How long have you used this product?

❑ Less than one month ❑ Less than six months ❑ Less than one year

❑ 1 to 2 years ❑ More than 2 years

How much have you read of this manual?

❑ Entire manual ❑ Specific chapters ❑ Used only for reference

	Agree				*Disagree*
The software was fully and accurately described	❑	❑	❑	❑	❑
The manual was well organized	❑	❑	❑	❑	❑
The writing was at an appropriate technical level (neither too complicated nor too simple)	❑	❑	❑	❑	❑
It was easy to find the information I was looking for	❑	❑	❑	❑	❑
Examples were clear and easy to follow	❑	❑	❑	❑	❑
Illustrations added to my understanding of the software	❑	❑	❑	❑	❑
I liked the page design of the manual	❑	❑	❑	❑	❑

If you have specific comments or if you have found specific inaccuracies, please report these on the back of this form or on a separate sheet of paper. In the case of inaccuracies, please list the relevant page number.

May we contact you further about how to improve SCO documentation? If so, please supply the following details:

Name _____ *Position* _____

Company _____

Address _____

City & Post/Zip Code _____

Country _____

Telephone _____ *Facsimile* _____

1 February 1993

BH01207P001

SCO® UNIX® Operating System
System Administrator's Guide

SCO®