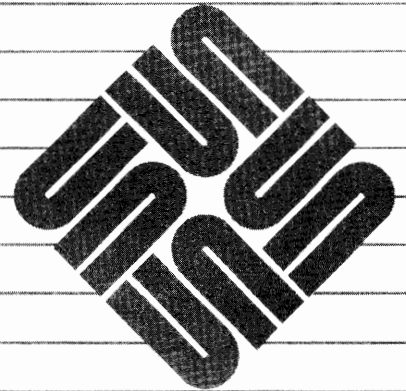




System & Network Administration



The Sun logo, Sun Microsystems, Sun Workstation, NeWS, and TOPS are registered trademarks of Sun Microsystems, Inc.

Sun, Sun-3, Sun-4, SunInstall, SunOS, SunView, ALM, NSE, NFS, and SPARC are trademarks of Sun Microsystems, Inc.

UNIX is a registered trademark of AT&T.

All other products or services mentioned in this document are identified by the trademarks or service marks of their respective companies or organizations, and Sun Microsystems, Inc. disclaims any responsibility for specifying which marks are owned by which companies or organizations.

Chapters 23 through 25 in the final section of this manual were originally derived from the following works:

Fsck — The UNIX File System Check Program

was originally written by T. J. Kowalski, Bell Laboratories, Murray Hill, New Jersey; and was revised by Marshall Kirk McKusick, Computer Systems Research Group, University of California at Berkeley.

Sendmail — An Internetwork Mail Router

was originally written by Eric Allman, formerly of Project Ingres, University of California at Berkeley.

Sendmail — Installation and Operation Guide

was originally written by Eric Allman, formerly of Project Ingres, University of California at Berkeley.

LEGAL NOTICE TO USERS: Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications plc., and may also be a trademark of various telephone companies around the world. Sun will be revising future versions of software and documentation to remove references to Yellow Pages.

Copyright © 1990 Sun Microsystems, Inc. – Printed in U.S.A.

All rights reserved. No part of this work covered by copyright hereon may be reproduced in any form or by any means – graphic, electronic, or mechanical – including photocopying, recording, taping, or storage in an information retrieval system, without the prior written permission of the copyright owner.

Restricted rights legend: use, duplication, or disclosure by the U.S. government is subject to restrictions set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 52.227-7013 and in similar clauses in the FAR and NASA FAR Supplement.

The Sun Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees.

This product is protected by one or more of the following U.S. patents: 4,777,485 4,688,190 4,527,232 4,745,407 4,679,014 4,435,792 4,719,569 4,550,368 in addition to foreign patents and applications pending.

Contents

Chapter 1	The System Administrator's Role	3
1.1.	System Administration--The Big Picture	3
	Sun Architecture Types	3
	Sun Equipment Configurations	3
	The System Administrator's Tasks	5
	Road Map through System Administration Documentation	5
1.2.	Developing Your Administration Procedures	8
Chapter 2	Introducing the SunOS Operating System	11
2.1.	Introducing the SunOS and UNIX Operating Systems	11
	How Does Your Machine Get Its Kernel?	12
	What Does the Kernel Do?	12
	What's in the Rest of the Operating System	13
2.2.	Becoming Superuser	13
	Assigning a Password to the Root User Name	13
	Logging in as Superuser from a Shell	14
2.3.	Using the <i>SunOS Reference Manual</i>	14
	Displaying an Online Man Page	15
2.4.	The SunOS File Systems	16
	How Your Machine Gets its File Systems	16
	File Systems on a Client	18
	File Systems on a File Server	18
	Swap Space	18
	Importance of Symbolic Links	19

The root File System	20
The /var Directory Hierarchy	21
The usr File System	22
The /usr/kvm Directory and Kernel Architectures	25
The export File System	27
The /export/exec Directory Hierarchy	27
Other Directories in /export	31
The home Directory Tree	31
The /etc and /usr/etc Directories	32
Additional File Systems Supported by Release 4.1	34
Setting Up the tmpfs Memory-based File System	34
Who Should Use tmpfs File Systems?	35
Suggestions for Using tmpfs	35
Procedures for Setting Up tmpfs	36
2.5. Major System Administration Databases and Files	37
Files for Administering Diskless Clients	37
crontab	37
fstab	38
passwd	38
group	39
hosts	40
hosts.equiv	40
.rhosts	40
aliases	41
printcap	41
Files for Administering a Dataless Client	42
Files for Administering a Standalone System	42
Files for Administering a Server	43
dumpdates	44
ttytab	44
install	45
bootparams	46
exports	46

hosts	47
fstab	47
ethers	48
networks	48
netgroup	48
Chapter 3 Introducing Networks	51
3.1. Performing Network Administration Tasks	51
3.2. Types of Networks	52
Local Area Networks	52
Wide Area Networks	53
uucp	53
The Internet and TCP/IP	53
3.3. Entities on a Network	54
Communications Media	54
Server	54
Client	54
Modem	54
Network Services	54
3.4. The Concept of Administrative Domains	55
3.5. Obtaining an IP Address	56
Chapter 4 Glossary of Terms	57
Part Two: System Administration Procedures	65
Chapter 5 Booting Up and Shutting Down Your System	67
5.1. Changes to /boot Prior to 4.x Releases	67
5.2. Powering Up Self-Test Procedures	68
5.3. The Automatic Boot Process	69
Booting from a Local Disk	69
Automatic Booting from Disk	69
Booting over the Network	70
Booting a New Client over the Network	70

The TFTP Boot Process	70
The bootparamd Daemon and /etc/bootparams File	71
5.4. Aborting a Booting Sequence	73
5.5. Booting from Specific Devices	73
The Standard Booting Syntax	74
Booting Up in Single-User Mode	74
5.6. The <code>init</code> Daemon and System Initialization Scripts	75
5.7. Shutdown Procedures	76
The <code>shutdown</code> Command	76
The <code>halt</code> Command	76
The <code>reboot</code> Command	77
The <code>fastboot</code> and <code>fasthalt</code> Commands	77
Chapter 6 File Maintenance	79
6.1. Backing Up File Systems	79
File Systems That Should Be Backed Up	79
Who Needs to Back Up?	80
File Systems to Dump on a Standalone	80
File Systems to Dump on a Server	80
Planning a Dump Strategy	81
Important Planning Considerations	81
Backup Strategies for a Server	83
Backup Strategies for a Standalone	85
Tapes and Equipment Used for Backup	86
Specifications for 1/2 Inch Tapes	87
Specifications for 1/4 Inch Tapes	88
Using the <code>dump</code> Command	89
<code>dump</code> Command Syntax	89
Using Dump to Archive Files	91
Planning Your Archiving Strategy	92
Procedures for Dumping Your Configurations	92
Procedures for Dumping to Half Inch Tapes	93

Procedures for Dumping to Quarter-Inch Tapes	95
Remote File System Dumps Over the Local Area Network	99
6.2. Restoring Files and File Systems	101
Using the <code>/etc/dumpdates</code> File	101
Using the <code>restore</code> Command	102
Restoring Individual Files	103
Procedures for Restoring a File	104
Restoring Files Over a Local Area Network	107
Restoring an Entire File System	108
The Strategy to Restore a Whole File System	108
The <code>umount</code> Command	109
The <code>newfs</code> Command	109
Mounting a Local File System	110
Determining Which Tapes to Use	110
Procedures for Restoring an Entire File System	111
Restoring a Damaged / or <code>/usr</code> File System	112
Procedures for Restoring <code>root</code>	112
Restoring the <code>/usr</code> File System	114
6.3. Monitoring File System Usage	115
Checking File System Usage	115
<code>ls</code> and File System Usage	115
The <code>du</code> Command	116
The <code>quot</code> Command	117
The <code>find</code> Command	117
Checking Disk Usage	119
Making Room on File Systems	119
6.4. The Disk Quota System	120
Setting Up the Quota System	121
Turning on the Quota System	122
Administering the Quota System	122
The User's View of Disk Quotas	123
What To Do When Quota Limit Is Reached	123
Quotas and Clients	124

Chapter 7 Administering Security	125
7.1. Security on Your Machine--an Overview	125
Protecting System Integrity through Security Barriers	126
System Security Barriers	126
Data Security Barriers	127
7.2. Protecting Your Local Machine	128
Password Security and the <code>/etc/passwd</code> File	128
The <code>passwd</code> File and NIS	130
Procedures for Setting Up a New User's Password	131
The <code>passwd</code> File and NIS	132
Setting Up User Groups on the Local Machine	132
The <code>/etc/group</code> File	133
Determining Group Membership	134
Creating New User Groups	134
Changing a User's Group ID	135
7.3. Protecting Files on the Local Machine	135
How File Permissions Are Stored	136
Changing File Permissions with <code>chmod</code>	137
Absolute Mode Made Easy	139
Displaying File Permissions with <code>ls -l</code>	140
Changing Ownership of a File	141
Changing Group Ownership of a File	142
Assigning Permissions at File Creation with <code>umask</code>	142
Encrypting Files	142
DES Encryption	143
7.4. Setting Security Measures for Executing Programs	143
Setting the Sticky Bit	144
Real and Effective User and Group IDs	145
Setting the User ID	145
Setting the Group ID	146
System Programs That Are <code>setgid</code>	147
How <code>setgid</code> Affects Directories	147
How SunOS Commands Affect User and Group IDs	148

7.5. The Administrator's Guide to Safe Systems	149
Security Administration	149
Password Administration	150
Safeguarding the Superuser Privileges	150
A Rogue's Gallery of Computer Invaders	152
Invasion by Trojan Horses	152
Trojan Mules	153
Computer Viruses	153
Computer Worms	153
Keeping an Individual Account Secure	153
Safeguarding Initialization Files	154
Monitoring the <code>.rhosts</code> File	154
Monitoring Security Through Options to <code>ls</code>	154
Creating a Safe Search Path	155
Temporary Directory Awareness	155
Protecting Unattended Workstations	155
Keeping File Systems Secure	155
Ensuring Device Security	155
Controlling <code>setuid</code> Programs	156
Mounting and Unmounting File Systems	159
Keeping System Directories and Files Secure	160
Protecting the <code>/etc/passwd</code> File	161
Administering Password Aging	161
Protecting the <code>/etc/group</code> File	164
Protecting the <code>/etc/aliases</code> File	164
Protecting <code>/var/spool/cron</code>	164
Promoting Physical Security and User Awareness	165
Promoting User Awareness	165
Administrator Awareness	166
Suggestions for Keeping Systems Secure	166
What if Security is Broken?	167
Chapter 8 Administering Workstations	169

8.1. Setting Up a Workstation's Operating Environment	169
Customizing the Administrative Databases	169
Step 1: Creating Home Directories	170
Step 2: Set Up Password Security	171
Step 3: Setting Up File Sharing	171
Step 4: Setting Up User Groups	172
Step 5: Enabling Printing	172
Step 6: Enabling Electronic Mail	172
Step 7: Ensuring Network Security	172
8.2. Adding Diskless Workstations and Software	173
Using the <code>add_client</code> Utility	173
<code>add_client</code> Examples	175
Using the <code>add_services</code> Utility	178
<code>add_services</code> Examples	179
Using <code>rm_client</code>	185
8.3. System Crashes	186
Standard Machine Behavior after a Power Loss	186
System Crash Dumps	187
Crash Error Messages	187
Enabling Crash Dumps	187
Analyzing System Crash Dumps	188
User Program Crashes	188
Crashes Associated with Shared Libraries	189
Error Messages Associated with Shared Libraries	190
Recovering from Problems with Shared Libraries	190
Installing a New Shared Library	190
When To Call Sun for Help	191
8.4. Making Local Modifications	191
8.5. Setting Up and Maintaining <code>crontab</code>	191
Using the <code>crontab</code> Editor	192
8.6. System Log Configuration	193
Possible Error Message Sources	195
Pre-4.0 Program Logging to <code>syslog</code> Daemon	196

4.1 Program Logging to Pre-4.0 syslog Daemon	196
8.7. Accounting	197
What Accounting Is	198
Accounting Programs	198
Setting Up Accounting	198
Daily Accounting	199
The runacct Program	200
Re-entrant States of the runacct Script	200
runacct Error Messages	202
Files Produced by runacct	203
Fixing Corrupted Files	203
Fixing wtmp Errors	203
Fixing tacct Errors	204
Restarting runacct	204
Billing Users	205
Setting Up Non-Prime Time Discounts	205
Daily Accounting Reports	206
Daily Report	206
Daily Usage Report	207
Daily Command Summary	209
Total Command Summary	211
Last Login Report	212
Looking at the pacct File with acctcom	213
Accounting Files	215
Chapter 9 Reconfiguring the System Kernel	219
9.1. Why Reconfigure the Kernel?	219
9.2. Parts of a Kernel Configuration File	220
The Reconfiguration Process	220
Major Sections of the GENERIC Configuration File	220
The System Identification Section	223
General System Description Lines	223
System-Specific Description Lines	225

The Pseudo-Devices Section	226
The Connections Section	226
The Devices Section	227
Device Description Lines	227
9.3. Modifying the Kernel Configuration Files	229
Modification Procedures	229
The Sun-3 GENERIC Kernel	232
9.4. How to Reconfigure the Kernel	242
Kernel Reconfiguration Procedures	242
Background Information	243
Method 1	243
Standalone or Server	244
Diskless Client	244
Dataless Client	244
Method 2	244
9.5. Changing Swap Space	246
Procedures for Increasing Swap Space	246
Setting Up a Second Swap Partition	247
Minimum Swap Space	248
Chapter 10 Maintaining Disks	249
10.1. An Overview of <code>format</code>	249
Some of the <code>format</code> Pitfalls, and How to Avoid Them	250
10.2. Interacting with <code>format</code>	251
Types of Input	251
Numbers	251
Block Numbers	251
Command Names	252
Other Names	252
Supporting Features in <code>format</code>	252
Help Facility	252
Maximum Operator	252
Defaults	253

10.3. Running format	253
-f <i>command_file</i>	253
-l <i>log_file</i>	253
-x <i>data_file</i>	253
-d <i>disk_name</i>	253
-t <i>disk_type</i>	253
-p <i>partition_name</i>	254
-s	254
10.4. Using format for Basic Maintenance	254
Formatting a New Disk from Sun	254
Formatting a New Disk from Another Source	257
Setting up or Changing Disk Partitions	262
Repairing a Defective Sector	264
Relabeling a Corrupted Disk	268
Creating a Defect List	270
10.5. The format.dat File	273
Search Path	275
Disk Type	275
Partition Tables	277
10.6. format Command Reference	278
The <i>Command Menu</i>	278
The <i>Partition Menu</i>	286
The <i>Analyze Menu</i>	289
The <i>Defect Menu</i>	293
10.7. format Error Messages	301
Errors for both SMD and SCSI Drives	301
Errors for SCSI Drives	311
Chapter 11 Adding Hardware to Your System	315
11.1. Adding a Board to Your System	315
Kernel Modification	315
Kernel Configuration for Servers	316
Adding Devices	317

Trouble Shooting	320
11.2. Connecting Devices to Asynchronous Serial Ports	320
General Theory	320
General Debugging Hints	322
11.3. Adding a Terminal to Your System	322
Serial Port and Cable Connection	322
Kernel Modification	323
Adding Devices	323
Problems	326
11.4. Adding a Modem to Your System	326
Cable Connection and Modem Switches	326
The Hayes Smartmodem 1200	327
The USRobotics Courier 2400	327
Telebit TrailBlazer	328
Settings for other modems	328
Kernel Modification	328
System Modification	329
Using mknod	329
Updating ttytab	330
Kernel Modification for Boards Not Supporting ttysoftcar	331
System Modification	334
Using mknod	334
Updating ttytab	335
The /etc/remote File	337
Hayes Specific Considerations	337
Problems	338
11.5. Adding a Printer to Your System	340
Serial Port and Cable Connection	340
File Modifications	341
Hooking Up a Printer to a VPC-2200 Multibus Board	342
Card Cage Installation And Cable Hook-up	342
Kernel Modification	342

File Modification	344
Chapter 12 Maintaining Printers and Print Servers	345
12.1. Introduction	345
12.2. Overview	345
12.3. Commands	346
12.4. Setting Up	346
The /etc/printcap File	347
Printers on Serial Lines	348
Printers on Parallel Ports	349
Remote Printers	349
12.5. Output Filters	350
Output Filter Specifications	351
12.6. Access Control	352
12.7. Line Printer Administration	353
Entering Commands and Parameters	353
Abbreviations	353
Obtaining Help	354
Restarting a Printer	354
Determining When a Printer Daemon Has Died	354
Exiting from lpc	354
Other Printer Administration Commands	354
Enabling and Disabling printer queues	355
Stopping and Starting Printer Daemons	355
Notifying users of a Disabled Printer	355
Aborting a Printer Daemon	355
Adjusting the Printer Queue	355
12.8. Troubleshooting	356
lpr	356
lpq	357
lprm	358
lpd	358
lpc	358

Part Three: Network and Communications Administration	359
Chapter 13 The SunOS Network Environment	361
13.1. Introducing the Internet Protocol Family	361
The Internet Protocol Suite	362
TCP/IP Protocol Structure	362
Administration tasks	366
13.2. Setting Up Network Software	366
IP Addressing	367
Addressing and Network Classification	367
IP Address Representation	367
Network Classes	367
Obtaining an IP Network Number	368
Creating IP Addresses for Your Hosts	369
Establishing a Domain	369
Selecting the Domain Name	369
Registering Your Domain	370
Setting a Host's Domain Name	370
Booting TCP/IP on Your Network Hosts	371
13.3. Maintaining TCP/IP-Related Files	371
The <code>hosts</code> Database	372
Adding IP Addresses to the <code>hosts</code> Database	372
The <code>ethers</code> Database	373
The <code>networks</code> Database	373
The <code>protocols</code> Database	374
The <code>services</code> Database	375
13.4. Security in a TCP/IP Environment	376
The <code>/etc/hosts.equiv</code> File	376
The <code>.rhosts</code> File	376
The <code>netgroup</code> database	377
How the Administrative Files Affect Network Security	377
Other Security Issues	379

13.5. Expanding Your Network	379
Hardware on a TCP/IP-based Network	379
Hardware Devices for Expanding the Local Network	380
Creating an Internetwork	381
Configuring a Router	382
Setting Up Subnets	384
Network Masks	384
The <code>netmasks</code> database	386
Changing from a Non-subnetted to a Subnetted Network	386
Examples of Subnets	387
13.6. Diagnosing Network Problems	387
The <code>ping</code> Command	387
The <code>ifconfig</code> Command	388
The <code>netstat</code> Command	389
Displaying Per Protocol Statistics	389
Displaying Communications Controller Status	391
Displaying Routing Table Status	391
Displaying Routing Statistics	391
Software Checks	392
<code>rwhod</code> and <code>routed</code>	392
Logging Network Problems	393
Chapter 14 The Sun Network File System Service	395
What is NFS	395
14.1. How NFS Works-an Overview	396
File Hierarchies	396
Servers and Clients	398
Servers and Exporting	398
Clients and Mounting	399
14.2. Setting Up and Maintaining an NFS Server	399
Setting Up a Server	399
Maintaining a Server	400
Exporting Directories	401

The /etc/exports File	401
Editing /etc/exports	402
Explicitly Exporting Directories with <code>exportfs</code>	403
Unexporting Hierarchies	404
Explicitly Unexporting Hierarchies with <code>exportfs</code>	404
The /etc/xtab File	405
Upgrading an NFS Server from Homogeneous to Heterogeneous	405
14.3. Setting Up and Maintaining an NFS Client	405
Setting Up a Client	405
Maintaining a Client	405
Mounting Files from an NFS Server	406
An NFS Client's <code>fstab</code> File	406
Making Mount Points	407
Mounting and Unmounting File Hierarchies	407
Explicitly Mounting Hierarchies with <code>mount</code>	408
Explicitly Unmounting Hierarchies with <code>umount</code>	410
14.4. Obtaining Information	411
<code>exportfs</code>	411
<code>showmount</code>	411
<code>mount</code>	411
<code>showfh</code>	412
14.5. Handling NFS Problems	412
Determining Where NFS Service Has Failed	413
Debugging Hints	415
Server Problems	416
Clearing Remote Mounting Problems	418
Error Messages Related to Remote Mounts	418
When <code>mount</code> Hangs Without an Error Message	421
Fixing Hung Programs	421
Fixing a Machine that Hangs Part Way Through Boot	422
Speeding Up Slow Access Times	422
14.6. Making the Network More Secure	423

Allowing Root Access over the Network	423
Privileged Ports	424
The Lock Manager	424
Synchronizing the time	425
14.7. Secure NFS	425
Security Shortcomings of NFS	425
14.8. Administering Secure NFS	426
14.9. RPC Authentication	429
UNIX Authentication	429
DES Authentication	429
Public Key Encryption	431
Naming of Network Entities	432
Applications of DES Authentication	433
Security Issues Remaining	434
Performance	435
Problems with Booting and <code>setuid</code> Programs	436
Conclusion	437
14.10. Secure NFS References	437
Chapter 15 Using the NFS Automounter	439
15.1. The Automounter	439
How the Automounter Works	439
15.2. Preparing the Maps	440
The Master Map	441
Direct and Indirect Maps	442
Writing a Master Map	443
Mount point /-	443
Mount point /home	443
Mount point /net	443
Writing an Indirect Map	445
Writing a Direct Map	446
Multiple Mounts	447
Multiple Locations	450

Specifying Subdirectories	451
Metacharacters	452
Ampersand (&)	452
Asterisk (*)	453
Backslash (\)	454
Double quotes (~)	455
Environment Variables	455
Including Other Maps	456
15.3. Starting automount	457
The Temporary Mount Point	459
The Mount Table	460
Modifying the Maps	460
Modifying the Master Map	460
Modifying Indirect Maps	460
Modifying Direct Maps	460
Mount Point Conflicts	461
15.4. Problem Solving	461
automount Sequence of Events	462
Error Messages Related to automount	464
Error Messages Generated by the Verbose Option	464
General Error Messages	465
Chapter 16 The Network Information Service	469
16.1. What is NIS	469
The NIS elements	469
16.2. The NIS Environment	470
The NIS Domain	471
NIS Machine Types	471
NIS servers	471
NIS Clients	472
NIS Binding	472
ypbind and Subnetting	473
NIS Maps	473

16.3. Deploying the NIS Service	475
Establishing the Domain	475
Preparing Network Databases for NIS Service	476
Preparing Files on NIS Clients	476
Preparing <code>hosts</code>	477
Preparing <code>passwd</code>	477
Preparing <code>group</code>	478
Preparing Network Databases on the Master Server	478
The <code>publickey</code> map	479
Other Maps	480
Making the maps	480
The default Makefile	480
Setting Up the Master Server	482
Creating the Master Server	483
Setting Up Slave Servers	484
Setting Up NIS clients	485
Heterogenous Slaves	486
Modifying Default Maps	487
Propagating an NIS Map	487
Using <code>crontab</code> with <code>ypxfr</code>	487
Using Shell Scripts with <code>ypxfr</code>	488
Directly Invoking <code>ypxfr</code>	489
Logging <code>ypxfr</code> 's Activities	489
16.4. Administering Users on an NIS Network	489
How NIS Maps Affect Security	489
Changing the NIS Password	490
How Netgroups Affect Security on an NIS Network	491
Adding a New User to an NIS Domain	491
Making the Home Directory	492
16.5. Obtaining Map Information	492
16.6. Advanced NIS Administration	493
Updating Existing Maps	493
Modifying the Makefile	494

Creating and Modifying Non-Default Maps	495
Updating Maps Built from Existing Text Files	495
Updating Maps Built from Standard Input	496
Adding New NIS Maps to the Makefile	497
Adding a New NIS Server to the Original Set	497
Changing a Map's Master Server	498
Using NIS in conjunction with DNS	499
Problems in Heterogeneous NIS Domains	500
16.7. Summary of NIS Related Commands	501
16.8. Fixing NIS Problems	502
Debugging an NIS Client	502
Hanging Commands on the Client	502
NIS Service is Unavailable	504
ypbind Crashes	505
ypwhich Displays are Inconsistent	506
Debugging an NIS Server	506
Servers Have Different Versions of an NIS Map	506
ypserv Crashes	507
16.9. Turning Off NIS Services	508
16.10. Default NIS Maps	509
Chapter 17 Administering Domain Name Service	513
17.1. What is DNS?	513
17.2. The DNS Domain Hierarchy	513
The Root Level Domain	515
Top Level Domains	516
Second Level Domains	516
Local Administrative Domains	516
17.3. Name Space and Administrative Zones	516
How Name Space Relates to Host and Domain Names	517
How Name Space Relates to Zones	518
Name Space and the IN-ADDR.ARPA Domain	518
DNS Servers and Clients	519

Clients	519
Master Servers	519
Caching and Caching-Only Servers	520
Forwarding Servers	520
17.4. Name Server Files	520
Resolver Configuration File	521
Boot File	521
Boot File For a Primary Server	522
Unused lines	525
Boot File For a Secondary Server	525
Boot File for Primary and Secondary Server	525
Boot File for Caching-Only Server	526
Boot file for Forwarding Server	526
17.5. Standard Resource Record Format	526
Special characters	527
Control entries	528
Resource Record Types	528
SOA - Start Of Authority	530
NS - Name Server	531
A - Address	531
HINFO - Host Information	532
WKS - Well Known Services	532
CNAME - Canonical Name	533
PTR - Domain Name Pointer	533
MX - Mail Exchanger	534
MB - Mailbox	534
MR - Mail Rename Record	535
MINFO - Mailbox Information	535
MG - Mail Group Member	536
17.6. A Practical Example	536
Adding a Cache Only Server	545
Self-contained DNS	545
17.7. Setting Up DNS	547

Starting named	547
Starting the resolver	547
17.8. Modifying the database	548
named's PID	548
Reload (SIGHUP)	548
17.9. Debugging named	548
Database Browsing (SIGINT)	548
Turning on debugging (SIGUSR1)	549
Turning off debugging (SIGUSR2)	549
Using nslookup	549
17.10. Administering DNS for Your Domain	550
Types of Administrators and Their Responsibilities	550
17.11. Acknowledgements	552
17.12. References	553
Chapter 18 The Remote File Sharing Service	555
18.1. RFS in the SunOS Network Environment-an Overview	555
RFS File Servers and Resource Sharing	556
RFS Clients and Mounting	557
How Resource Sharing Takes Place	558
The RFS Name Server and its Functions	559
The RFS Domain	560
Your Role as an RFS System Administrator	561
18.2. Installing RFS	562
Installing RFS on a Machine with a Disk	562
Loading RFS from an NFS File Server to Its Clients	563
18.3. Setting Up the RFS Domain	564
Setting Up the RFS Name Server	564
Defining the RFS Domain Name	565
Setting Up the <code>rfmaster</code> File	565
Running <code>dorfs</code>	567
RFS Files and Daemons	568
Setting Up a Secondary RFS Name Server	568

Setting Up Secondaries for a New RFS Domain	568
Setting Up Secondaries for an Existing RFS Domain	569
Setting Up an RFS File Server	570
Procedures for Starting RFS on the File Server	570
Starting RFS on a Client	571
Starting RFS at Boot Time	572
18.4. Advertising Resources from an RFS File Server	573
Naming Your Resources	573
Checking the Advertise Tables	573
Using the <code>adv</code> Command	574
Advertising Devices as Resources	576
The <code>rstab</code> File	577
Aliases	578
Maintaining Resources	578
Maintaining Resource Security	578
Monitoring Resource Usage	579
Unadvertising Resources	579
18.5. Mounting Resources	580
Using <code>mount</code> to Mount RFS Resources	581
Mounting Resources During Booting	582
Mounting Limitations for RFS Clients and File Servers	583
Mounting Remote Devices	584
Mounting Guidelines for RFS File Servers	585
Unmounting Resources	585
Unmounting Resources on an RFS Client	585
Unmounting Advertised Resources on an RFS File Server	586
18.6. Setting Up RFS Security	587
RFS Security Levels—an Overview	587
Connect Security	587
Mount Security	588
User and Group Security	588
Determining Security Measures for Your Machine	589
Setting Up Connect Security for the RFS Domain	590

Procedures for the Primary RFS Name Server	591
Setting Up Password Verification on the File Server	593
Procedures for the RFS Client	595
Setting Up Mount Security	596
Setting Up User and Group Security	596
How Remote User Mapping Works	597
Defining Mapping Rules	598
The <code>idload</code> Command and Mapping	600
Choosing Mapping Strategy	601
<code>map name:name</code>	603
18.7. Maintaining RFS Services	603
Maintaining RFS for RFS File Server and Client Machines	604
Starting Up a Client	604
Running <code>dorfs stop</code> to Shut Down RFS	605
Changing the RFS Password	605
Updating User and Group Information	605
18.8. Domain Maintenance	606
Adding New Machines to <code>domain/passwd</code>	606
Removing a Machine from <code>domain/passwd</code>	606
Change Current RFS Name Server	607
Reassigning RFS Name Service	607
Adding New RFS Domains	607
Updating User and Group Files	607
18.9. Recovery in the RFS Environment	608
Primary Goes Down	608
Primary and Secondaries Go Down	608
RFS File Server Goes Down	609
<code>rfudaemon</code>	609
<code>rfuadmin</code>	609
Chapter 19 Administering C2 Security	611
19.1. Introduction	611
19.2. What is C2 security?	612

Levels of Security	612
19.3. Setting Up C2 Security	613
C2 Required Programs	613
Loading the C2 programs	615
Kernel Requirements	615
NIS Domains	615
Converting to C2	615
19.4. Password and Group Security	615
19.5. Definition of Auditing Terms	616
19.6. Running C2conv	618
A Summary of C2conv Actions	619
Before Running C2conv	619
19.7. Sample Run of C2conv	620
19.8. After Running C2conv	628
19.9. Changing the Audit State	631
Changing the System Audit State	631
Changing the User Audit State	631
Permanent User Audit State	631
Immediate User Audit State	632
Changing the Audit File	632
19.10. Looking at the Audit Trail	633
Static Examination	633
Watching on the Fly	633
Stopping Auditing	634
19.11. When Audit File systems Are Full	634
19.12. Running C2unconv	634
Chapter 20 Administering Electronic Mail	635
20.1. An Overview of the Electronic Mail System	635
Domain Names and sendmail	637
20.2. Configuring Machines for Mail Operation	637
Setting Up an NFS Mailbox Server and Its Clients	638
Converting Clients to Use Mailbox Servers	639

Setting Up the Mailbox Server Alias	639
Setting Up the Mailhost and Subsidiary Machines	640
Configuring Subsidiary Machines	640
Configuring the Mailhost	642
Establishing the Relay Host	642
Setting Up the Postmaster Alias	644
The <code>/etc/aliases</code> File	644
Handling Undelivered Mail	645
Special Considerations for Networks with NIS	646
Using Inverted NIS Domain Names	646
Mail and the Internet Domain Name Server	647
Testing Your Mail Configuration	647
Diagnosing Problems with Mail Delivery	648
The System Log	649
Format	649
Levels	649
Chapter 21 Administering the UUCP System	651
21.1. Introduction	651
21.2. UUCP Hardware	651
21.3. UUCP Programs	652
User Programs	652
Administrative Programs	653
Daemons	653
21.4. Starting UUCP	654
<code>uudemon.poll</code>	654
<code>uudemon.hour</code>	654
<code>uudemon.admin</code>	655
<code>uudemon.cleanup</code>	655
21.5. Supporting Data Base	655
21.6. Devices File	656
Protocols	660
21.7. Dialers File	660

Hardware Flow Control	663
Setting Parity	663
21.8. Systems File	663
Hardware Flow Control	667
Setting Parity	668
21.9. Dialcodes File	668
Correspondences	668
21.10. Permissions File	669
How Entries are Structured	669
Considerations	670
REQUEST Option	670
SENDFILES Option	670
MYNAME Option	671
READ and WRITE Options	671
NOREAD and NOWRITE Options	672
CALLBACK Option	672
COMMANDS Option	672
VALIDATE Option	673
MACHINE Entry for ~OTHER~	675
Combining MACHINE and LOGNAME	675
Forwarding	675
21.11. Poll File	676
21.12. Sysfiles File	676
21.13. Other UUCP Files	677
21.14. Administrative Files	677
21.15. Running UUCP over TCP/IP	679
21.16. Set Up Permissions File	680
21.17. Add uucp logins	680
21.18. Examples	681
Connecting Through a Modem - LUUCP	681
Connecting Through TCP/IP	681
Connecting Through a Direct Line - UUCP	682
Connecting Through a Direct Line - cu	682

Connecting Through a Modem - cu	682
21.19. UUCP Maintenance	682
Mail for UUCP	683
The Public Directory	683
21.20. UUCP Debugging	683
Check for Faulty ACU/Modem	683
Check Systems File	683
Debug Transmissions	683
Check Error Messages	684
Check Basic Information	685
21.21. This Version of UUCP	685
Data Files	685
Log and Status Files	685
Lock Files	685
Other Hidden Directories	685
Upgrading your UUCP system	685
Conversion Tips	686
21.22. UUCP Error Messages	687
UUCP ASSERT Error Messages	687
UUCP STATUS Error Messages	688
Part Four: Administrator's Reference	691
Chapter 22 Advanced Administration Topics	693
22.1. Files Used in Kernel Configuration	693
Adding Your Own Device Drivers	694
The <i>options optlist</i> Line	694
Adding Device Description Lines	694
22.2. Configuring Systems Without Source	695
22.3. Sharing Object Modules	695
Building Profiled Kernels	696
22.4. Rules for Defaulting System Devices	696
Configuration File Grammar	697
Lexical Conventions	699

Data Structure Sizing Rules	699
Compile Time Rules	699
22.5. Tuning IPC System Parameters	700
Why Reconfigure IPC Parameters?	701
IPC Message Parameters	702
IPC Semaphore Parameters	702
IPC Shared Memory Parameters	703
Named Pipe Parameters	704
22.6. TCP/IP Configuration Options for SunOS 4.1	704
Descriptions of TCP/IP Parameters	705
22.7. The Crash Kernel Debugger	707
Preserving Core Images	707
Common system panic messages, and their causes	707
The System Core Image	708
What Happens When a File System Is Nearly Full?	709
Starting crash	709
Useful crash Commands	709
What do I do now?	711
Chapter 23 Customizing sendmail Configuration Files	713
23.1. sendmail Features	714
23.2. sendmail Overview	715
Interfaces to the Outside World	715
Argument Vector/Exit Status	715
SMTP over Pipes	715
SMTP over a TCP Connection	715
How sendmail Works	715
Argument Processing and Address Parsing	715
Message Collection	716
Message Delivery	716
Queueing for Retransmission	716
Return to Sender	716
Message Header Editing	716

Configuration File	717
23.3. How to Implement and Use <code>sendmail</code>	717
Mail to Files and Programs	717
Message Collection	717
Message Delivery	718
Queued Messages	718
Configuration Overview	718
Macros	718
Header Declarations	718
Mailer Declarations	718
Name Rewriting Rules	719
Option Setting	719
23.4. Basic Installation	719
<code>sendmail</code> Files	719
<code>/usr/lib/sendmail</code>	719
<code>/usr/lib/sendmail.mx</code>	720
<code>/etc/sendmail.cf</code>	720
<code>/var/spool/mqueue</code>	720
<code>/etc/aliases*</code>	720
<code>/etc/rc.local</code>	721
<code>/usr/lib/sendmail.hf</code>	721
<code>/var/spool/mail</code>	721
23.5. Aliases, Forwarding, and Mailing Lists	721
Potential Problems	723
Mailing Lists	723
Inclusion	724
NIS Aliases	724
Update Policy	724
Naming Conventions	724
Potential Problems	724
Forwarding	725
The Mail Queue	725
Printing the Queue	725

Format of Queue Files	725
Forcing the Queue	727
23.6. Arguments	728
Queue Interval	728
Daemon Mode	728
Debugging	728
Trying a Different Configuration File	728
Quick Configuration Startup	729
23.7. Tuning	729
Time Values	729
Queue Interval	729
Read Timeouts	729
Message Timeouts	730
Delivery Mode	730
Load Limiting	730
Log Level	731
File Modes	731
To setuid or not to setuid?	731
Temporary File Modes	731
Should my Alias Database be Writable?	731
23.8. The Configuration File	732
Building a Configuration File from Scratch	732
Purpose of the Configuration File	732
Domains and Policies	733
How to Proceed	733
Testing the Rewriting Rules — the <code>-bt</code> Flag	733
A Sample <code>sendmail</code> Configuration File	734
Configuration File Syntax	741
D — Define Macro	741
C and F — Define Classes	741
O — Set Option	742
P — Precedence Definitions	742
T — Define Trusted Users	743

H — Define Header	743
Special Header Lines	743
Return-Receipt-To:	743
Errors-To:	743
To:	743
R and S — Rewriting Rules	743
M — Define Mailer	744
The Semantics	744
Special Macros, Conditionals	744
Special Classes	746
The Left Hand Side	746
The Right Hand Side	747
Semantics of Rewriting Rule Sets	748
The “error” Mailer	749
Semantics of Mailer Descriptions	749
23.9. Command Line Arguments	751
23.10. Configuration Options	752
23.11. Mailer Flags	754
Chapter 24 File System Check Program	757
24.1. Overview of the File System	757
Superblock	757
Summary Information	758
Cylinder Groups	758
Fragments	759
Updates to the File System	759
24.2. Fixing Corrupted File Systems	760
Detecting and Correcting Corruption	760
Superblock Checking	760
Free Block Checking	761
Checking the Inode State	761
Inode Links	761
Inode Data Size	762

Checking the Data Associated with an Inode	762
File System Connectivity	763
24.3. <code>fsck</code> Error Conditions	763
Conventions	763
Initialization	764
Phase 1 – Check Blocks and Sizes	768
Phase 1B: Rescan for More Dup's	771
Phase 2 – Check Pathnames	772
Phase 3 – Check Connectivity	778
Phase 4 – Check Reference Counts	781
Phase 5 – Check Cyl Groups	784
Cleanup	786
24.4. References	786
Chapter 25 Modifying the <code>termcap</code> File	789
25.1. Types of Capabilities	789
25.2. Preparing Descriptions	790
25.3. Basic Capabilities	791
25.4. Parameterized Strings	792
25.5. Cursor Motions	793
25.6. Area Clears	793
25.7. Insert/Delete Line	793
25.8. Insert/Delete Character	794
25.9. Highlighting, Underlining, and Visible Bells	795
25.10. Keypad	796
25.11. Tabs and Initialization	796
25.12. Delays	797
25.13. Miscellaneous	797
25.14. Glitches and Braindamage	799
25.15. Similar Terminals	799
Appendix A Special Booting Procedures	801
A.1. Booting from an Alternative Disk	801

A.2. Booting from an NFS-Mounted Partition	803
A.3. Booting from Tape	804
A.4. Booting an Alternate Kernel from the Default Device	804
Appendix B Error Messages from the Monitor and Boot Program	805
Appendix C Timezones	825
C.1. TIMEZONE NAME:	825
Appendix D The Orange Book	829
Glossary	829
Discretionary Access Control	830
Object Reuse	830
Identification and Authentication	831
Auditing	831
Auditing Superuser Activities	832
Auditing Versus Time and Disk Space	832
What Events Are Audited	833
System Architecture	833
System Integrity	834
Appendix E Format of Audit Records	835
E.1. Header and Data Fields	835
E.2. Audit Records for System Calls	836
E.3. Audit Records for Arbitrary Text	847
Appendix F IP Number Registration Form	853
Appendix G Internet Domain Registration Form	859
References	863

Tables

Table 1-1	What You Should Learn to Manage a Diskless Client	6
Table 1-2	What You Should Learn to Manage a Dataless Client	6
Table 1-3	What You Should Learn to Manage a Standalone Workstation	7
Table 1-4	What You Should Learn to Manage a File Server	7
Table 1-5	What You Should Learn to Manage a Network Domain	8
Table 6-1	Default File Systems on a Standalone	80
Table 6-2	Default File Systems on a Server	81
Table 6-3	Schedule of Backups for a Server	83
Table 6-4	An Alternative Backup Schedule for a Server	85
Table 6-5	Sun Tape Controllers	86
Table 6-6	Specifications for a Xylogics Tape Controller with Fujitsu Tape Unit	88
Table 6-7	Specifications for Tapemaster Controllers and Xylogics/CDC Combination	88
Table 6-8	Devices and Tapes on a SCSI Tape Controller	89
Table 6-9	Sun Disk Controllers	91
Table 7-1	Common Permission Modes for Files	140
Table 7-2	Common Permission Modes for Directories	140
Table 11-1	Sun Supported Tape Controller Boards	318
Table 11-2	Sun Supported Disk Controllers	318
Table 11-3	Sun Supported Terminal Multiplexor Devices	319

Table 11-4 Sun Supported Ethernet Controller Devices	319
Table 11-5 Sun Supported Printer Devices	319
Table 11-6 Sun Supported Graphics/Windows Devices	319
Table 11-7 Miscellaneous Sun Supported Devices	319
Table 11-8 Using MAKEDEV For New Boards	324
Table 13-1 TCP/IP Protocol Layers	362
Table 16-1 A Basic Set of NIS Maps	474
Table 16-2 NIS/DNS in Heterogeneous NIS Domains	501
Table 16-3 NIS Map Descriptions	509
Table 17-1 Commonly used types of RR's	528
Table 19-1 List of Required Programs	614
Table 19-2 List of Header Files	615
Table 19-3 List	618
Table 21-1 Format and example of Devices File	656
Table 21-2 Format and example of Dialers File	661
Table 21-3 Format and example of Systems File	664
Table 21-4 Format and example of Dialcodes File	668
Table 21-5 Ownership and Permissions of the new files	686
Table 21-6 UUCP ASSERT Error Messages	687
Table 21-7 UUCP STATUS Error Messages	689
Table 22-1 TCP/IP Default Parameters	705

Figures

Figure 2-1 Mounting the /usr/local Directory	17
Figure 2-2 A Symbolic Link	19
Figure 2-3 Directories with Executables on a Homogeneous Server	28
Figure 2-4 Directories with Executables on a Heterogeneous Server	30
Figure 7-1 System Security Barriers	127
Figure 7-2 Data Security Barriers	128
Figure 7-3 Format of Permission Bits	136
Figure 7-4 Octal Values for the Bits in a Triplet	138
Figure 7-5 Format of the Program Execution Triplet	144
Figure 8-1 A Sample crontab File	192
Figure 8-2 A sample /etc/syslog.conf file	193
Figure 8-3 Repairing a wtmp File	204
Figure 8-4 Repairing a tacct File	204
Figure 8-5 Sample Daily Report	206
Figure 8-6 Sample Daily Usage Report	208
Figure 8-7 Sample Daily Command Summary	209
Figure 8-8 Sample Total Command Summary	212
Figure 8-9 Sample Last Login	213
Figure 11-1 Communications Through Modems	321
Figure 11-2 ttytab for systems with boards supporting ttysoftcar	330
Figure 11-3 ttytab for systems with boards not supporting ttysoftcar	336

Figure 13-1 Sender/Receiver Interaction	363
Figure 13-2 Network Address Structure	368
Figure 13-3 rlogin/rcp/rsh Security	378
Figure 13-4 Routers and Networks	382
Figure 13-5 Network Mask	385
Figure 14-1 File Systems on a Disk	396
Figure 14-2 File Systems after mounting	396
Figure 14-3 File Systems on Separate Machines	397
Figure 14-4 File Hierarchies on Separate Machines after an NFS Mount	398
Figure 14-5 DES Authentication Protocol	430
Figure 15-1 Master, Direct, and Indirect Maps	441
Figure 15-2 Sample auto.master Map	443
Figure 15-3 Sample auto.home Map	445
Figure 15-4 Sample auto.direct Map	447
Figure 15-5 Example of a Multiple Mount	447
Figure 15-6 Example of a Hierarchical Multiple Mount	449
Figure 15-7 Example of <i>subdirectory</i> Field	452
Figure 15-8 Example of Use of Ampersand	453
Figure 15-9 Example of Use of Ampersand and Asterisk	453
Figure 15-10 Automount Disguised as NFS Server	462
Figure 15-11 Automount's Symbolic Link: Direct Map	463
Figure 15-12 Automount's Symbolic Link: Indirect Map	463
Figure 16-1 Master, slaves and clients	472
Figure 17-1 A Typical Internet Domain Hierarchy	515
Figure 17-2 Zones in the Internet Name Space	517
Figure 17-3 Boot file and data files	522
Figure 17-4 An imaginary network	537
Figure 18-1 Resource Sharing Between an RFS Client and File Server	558

Figure 18-2 Layout of Network Administrative Domain arts.com	561
Figure 18-3 Security Levels on Machines Using RFS	590
Figure 20-1 A Typical Electronic Mail Configuration	638
Figure 21-1 Correspondences Between the Files	669
Figure 23-1 Sendmail System Structure	714
Figure 23-2 How sendmail Uses Aliases	722
Figure 23-3 Rewriting Set Semantics	748

Preface

System and Network Administration is a resource manual for managing Sun computers running the SunOS Operating System, Release 4.1. It contains procedures for maintaining all types of Sun equipment configurations, mostly, but not entirely, from a software perspective. The text explains system administration and networking concepts for administrators at all levels of expertise.

This manual is a companion to *Installing SunOS 4.1*. It assumes that you have already installed Release 4.1 on new or already-in-use Sun computers, which you now are going to maintain and manage.

Types of Configurations

The manual assumes you manage one of the following basic Sun configurations:

- A workstation without a disk, also called a *diskless client*, that will rely on a network for essential services.
- A workstation with a disk, also called a *dataless client*, that will rely on a network for essential services.
- A *standalone* workstation with a disk and tape drive that does not depend on a network for essential services, however, you can attach it to a network.
- A *time-sharing* system, where one standalone computer, possibly with a printer, provides support for directly attached terminals or terminals communicating over telephone lines. This system may or may not be attached to a network.
- An individual *file server* machine sharing its services to clients on a network.
- A *network domain* with different types of servers and clients.

These configurations are explained more fully in Chapter 1, “The System Administrator’s Role,” and in *Installing SunOS 4.1*. They are repeated here to assist you in determining which parts of the *System and Network Administration* apply to you.

How to Use This Manual

System and Network Administration contains four parts, which address the needs of system administrators with various levels of experience. Each part opener explains which chapters you should read, depending on your prior experience and type of configuration you manage. The four parts are summarized below:

Note: Part One also contains a glossary of terms, designed for administrators at all levels of technical expertise.

- Part One, *System Administration for Beginners* explains basic concepts relevant to administering all types of Sun equipment configurations, including an introduction to the SunOS file systems and general networking concepts.

You should read the part if you are learning system administration for the first time. If you are an experienced system administrator but are new to Sun equipment and have never used a UNIX† based system, you may also want to read Part One.

- Part Two, *System Administration Procedures*, explains concepts and contains instructions for maintaining all types of Sun configurations. Topics include boot up and shutdown procedures, backing up and restoring file systems, crash recovery, reconfiguring the kernel, adding boards, terminals, and printers, and disk administration.

The text assumes that either you are an experienced system administrator—though not necessarily experienced with Sun equipment—or have read Part One of this manual. Part Two contains information for managing all Sun configurations.

- Part Three, *Network and Communications Administration*, explains how to set up and maintain a local area network running TCP/IP, the networking protocol provided by SunOS. The text also explains how to administer the network file system, yp name service, domain name service, remote file sharing system, and electronic mail. It also explains how to set up and administer the uucp communications service.

This part assumes that you understand the concepts and have used the procedures explained in the first two parts of the manual. Read Part Three if you are administering a network server. If you are administering a network client, you need to read the sections that apply to clients. If you are administering a standalone or time-sharing system, you don't have to read the information in this part, unless you want to attach the machine to a network or set up uucp.

- Part Four, *Administrator's Reference*, contains reference material, such as a detailed explanation of kernel configuration, the fsck program, the sendmail mail router, and the termcap file.

Part Four is intended for experienced administrators who want to tailor their systems to suit special needs, and for others who want to increase their knowledge of the software used for administering systems.

Supporting Documentation

If you need more information about topics in this manual, refer to the following:

- *Installing SunOS 4.1*
- *SunOS Reference Manual*

† UNIX is a registered trademark of AT&T.

- *Network Programming*
- *Writing Device Drivers* (for advanced administrators)

Documentation Conventions

The following conventions are used in the procedures and examples throughout this document:

- Prompts and error messages from the system are printed in `listing font like this`.
- Information that you type as a command or in response to prompts is shown in **boldface listing font like this**. Type everything shown in boldface exactly as it appears.
- Where parts of a command are shown in *italic text like this*, they refer to a variable that you have to substitute from a selection; it is up to you to make the proper substitution.
- Dialogues between you and the system are enclosed in gray boxes like the following:

```
host% ls personnel.rec
amina      azhar      bb          cameron
ernest     farrasha   gregorio   jim
linda     michael    stefania   susan
tony       turia
```

- Sections of program code are enclosed in clear boxes like the following:

```
int test[100];

main()
{
    register int a, b, c, d, e, f;

    test[a] = b & test[c & 0x1] & test[d & 0x1];
}
```

Part One: System Administration for Beginners

This part is designed to familiarize new system administrators with the tasks they will need to perform to administer a Sun computer. It also introduces SunOS system and networking concepts that are discussed in more detail in the remainder of the manual. Subjects covered in Part One include:

- The system administrator's tasks
- SunOS file system concepts and organization
- Introduction to networking and Sun network services

Part One concludes with a glossary of terms.

Who Should Read This Part

You should read Part One if you are:

- A novice system administrator, in which case you should read the entire part.
- An experienced system administrator who is now learning the UNIX† operating system, SunOS operating system, and Sun equipment, in which case you should read Chapters 2 and 3.

Prerequisite Knowledge

All procedures in this manual assume that you have read *Installing SunOS 4.1*, and have already installed this release using SunInstall program. The text also assumes that you have a basic familiarity with the SunOS operating system or UNIX system commands. If you are not familiar with these commands, the next chapters suggest additional reading materials.

Part One does not assume that you know anything about local area networks, the network file system, and electronic communications. If you see a network-related term that you don't understand, look the term up in the glossary in Chapter 4.

† UNIX is a registered trademark of AT&T.

The System Administrator's Role

This chapter briefly explains the types of tasks you will perform as a Sun system administrator, and possibly, network administrator. It points out basic knowledge you need for managing each type of configuration, then tells you where to find this information in this manual and others in your System Administration documentation set.

1.1. System Administration--The Big Picture

Sun Architecture Types

This section provides a generic overview of the equipment you will manage and the tasks you need to perform.

All Sun™ computers are classified by their architecture types. Each computer product will have two architecture types, an application architecture and a kernel architecture. The chapter *Introducing the SunOS Operating System* explains the term application architecture. The term *kernel architecture* refers to a machine's CPU chip and the kernel binary programs that it must run in order to function. The next table lists Sun computer products that run Release 4.1, the type of CPU chip resident on each product, and each product's kernel architecture type.

<i>Product Type</i>	<i>CPU Type</i>	<i>Architecture</i>
Sun-3™	Motorola 68020	sun3
Sun-3x	Motorola 68030	sun3x
Sun-4™	SPARC™	sun4
SPARCstation™	SPARC	sun4c

You can use the `arch -k` command to determine your machine's kernel architecture as shown next.

```
% arch -k
sun3x
```

Sun Equipment Configurations

System and Network Administration covers the common administration tasks you must perform to manage one or more of five categories of Sun equipment configurations. In a SunOS™ environment, any computer system that provides a network service, such as disk storage, file service, name service, or electronic mail is considered a *server*. Machines that receive these network services are called *clients*. These terms and other categories of servers are described in the chapter *Introducing Networks*. The following are the five types of configurations

you may administer.

□ *File server*

Any server that shares its disk storage and files—including those that certain client machines must access in order to operate—is called a *file server*. File servers are often described as either homogeneous or heterogeneous, depending on whether they serve one or more kernel architectures. A *homogeneous server* supports client machines with the same kernel architecture as it has. A *heterogeneous server* supports client machines that have both the same and different kernel architectures than the server. Additionally, a heterogeneous server may support equipment from manufacturers other than Sun—a feature of Sun’s “open systems” approach to networking.

Throughout *System and Network Administration*, the function of file servers is described as “providing *file service*.” File service refers to the file server’s activity when sharing its files with clients. All Sun file servers provide the Network File System (NFS™) file service. Additionally, Sun computers can provide the Remote File Sharing (RFS) service.

□ *Diskless client*

This is a computer that does not have its own disk. To operate fully, you must attach it to a local area network with at least one NFS file server. The diskless client relies on servers for its disk storage, and other services. Most significantly, the client relies on the NFS file server for the programs that enable it to boot.

Since diskless clients generally are used by one person, in most cases the user also functions as its system administrator.

□ *Dataless client*

This computer system has its own disk, on which you configure its local `root` and `swap` partitions. However, the machine receives the crucial `/usr` file system from an NFS file server; it cannot boot completely without the executable programs in `/usr`. Thus the client is considered “dataless.”

Like diskless clients, dataless clients are also primarily used by one person. The user is therefore also the system administrator.

□ *Standalone system*

A standalone has its own local disk with `/`, `/swap` and `/usr` partitions. It does not need an NFS file server to boot. You can operate the standalone as an independently functioning workstation with one user, attach it to a network, or use it as a time-sharing system with several users. If you attach the standalone to a network, it can use services provided by servers on the network, such as additional files, electronic mail, and the like.

□ *Time-sharing system*

This is a standalone system with terminals attached to its serial ports. It may also support terminals that connect over phone lines via a modem. The terminals rely on the standalone not only for disk storage, files, and printer

support, but also CPU time, since terminals do not have their own CPU. Large time-sharing systems, such as SPARCservers™, may additionally have directly connected workstations. The time-sharing system may or may not be on a network.

□ *Network domain*

A network domain is a large configuration consisting of some, perhaps even all, of the machine types listed above. The person in charge of the network domain is usually called the network administrator. Because network administration is a complex task, most network administrators are experienced system administrators with a certain amount of technical expertise.

If you need to learn more about the hardware of your particular configuration, refer to the hardware installation guides that are shipped with the equipment.

The System Administrator's Tasks

Your goal as administrator is to keep your equipment configuration running smoothly. To achieve this goal, you need to perform various tasks. Some of these tasks you may want to perform daily, such as doing incremental backups of the /home directory (if your configuration has a disk), or checking to see how full your file systems are (all configurations should do this one). Other tasks you may only perform once, such as adding your server or client machine to a network. Still others, such as restoring files after a crash, are tasks you hope you will never have to perform.

Following is a list of the most common system administration tasks, necessary on most configurations.

- Adding users
- Installing software, such as applications and operating system upgrades from Sun
- Installing hardware like additional boards, printers, terminals, and modems
- Maintaining the security and integrity of the system and the network
- Diagnosing and fixing software and hardware problems as they occur
- Checking file system use to make sure your file systems do not become full
- Maintaining printers, modems, and remote terminals
- Backing up file systems (unless you manage a diskless client)
- Maintaining network services, mail, and other communications services like uucp

Road Map through System Administration Documentation

This section contains tables that show you the crucial information that you should learn and procedures that you need to perform for your particular configuration. The tables also direct you to the chapters in the manual with this information.

Table 1-1 *What You Should Learn to Manage a Diskless Client*

Activity	Where It's Described
Understanding boot and shutdown	Chapter 5
Understanding the SunOS file systems	Chapter 2
Modifying <code>/etc/passwd</code>	Chapter 7
Modifying <code>/etc/fstab</code>	Chapter 14
Mounting file systems	Chapters 2,14,18
Where to make mount points	Chapter 14
Handling overloaded file systems	Chapter 6
Handling system crashes	Chapter 8
Adding and maintaining printers	Chapter 11
Setting up mail aliases	Chapter 20
Adding additional boards	Chapter 11

Table 1-2 *What You Should Learn to Manage a Dataless Client*

You should know everything in Table 1-1, plus the information in the following table:

Activity	Where It's Described
Maintaining disk subsystems	Chapter 10
Installing software	Chapter 8
Doing remote backups/restores	Chapter 6
Understanding NIS service	Chapter 16
Modifying <code>/etc/hosts.equiv</code>	Chapter 13

Table 1-3 *What You Should Learn to Manage a Standalone Workstation*

If you are managing a true standalone, referred to later as a non-networked standalone, you should learn the following.

<i>Activity</i>	<i>Where It's Described</i>
Understanding boot & shutdown	Chapter 5
Understanding the SunOS file systems	Chapter 2
Mounting local file systems	Chapter 2
Installing software	Chapter 8
Doing local backups/restores	Chapter 6
Maintaining disk subsystems	Chapter 10
Knowing files and commands in /etc	Chapter 2
Modifying permissions	Chapter 7
Adding local users and groups	Chapter 7
Handling overloaded file systems	Chapter 6
Adding and maintaining printers	Chapters 11 & 12
Adding additional boards	Chapter 11
Adding modems and terminals	Chapters 11 & 25
Fixing workstation problems	Chapter 8
Reconfiguring the Kernel	Chapter 9

If you are managing a standalone on a network, you should learn everything in the previous table, plus the following.

<i>Activity</i>	<i>Where It's Described</i>
Understanding networks	Chapter 13
Mounting remote file systems	Chapters 14 & 18
Modifying /etc/fstab	Chapter 14
Where to make mount points	Chapter 14
Using the Remote File Sharing Service	Chapter 18
Setting up mail aliases	Chapter 20
Understanding NIS service	Chapter 16
Modifying /etc/hosts.equiv	Chapter 13

Table 1-4 *What You Should Learn to Manage a File Server*

To manage a file server, you should learn everything in the tables for a standalone, plus the activities below.

<i>Activity</i>	<i>Where It's Described</i>
Setting up NFS	Chapter 14
Setting up the NFS automounter	Chapter 15
Setting up RFS	Chapter 18

Table 1-5 *What You Should Learn to Manage a Network Domain*

To manage a network domain, you should learn everything in the tables for a standalone and file server, plus the activities below.

<i>Activity</i>	<i>Where It's Described</i>
Setting up NIS service	Chapter 16
Setting up mail service	Chapter 20
Joining a wide area network	Chapter 13
Setting up subnets	Chapter 13
Setting up uucp	Chapter 21
Setting up Domain Name Service	Chapter 17

Although the tables list the basic knowledge necessary for maintaining your configuration, you probably don't want to limit yourself solely to this information. For example, if you manage a diskless client or non-networked standalone, most of the information in Part Three does not pertain to your configuration. However, you may want to read excerpts from this part to learn more about network administration. Once you have mastered the basics, you may then want to go on to the reference material in Part Four.

1.2. Developing Your Administration Procedures

As system administrator, you can make your work easier and get problems fixed more quickly by following the suggestions below. Remember, you should develop administrative procedures and standards to fit your particular configuration. It helps prevent rash action in a puzzling or unexpected situation.

Below is a suggested introductory checklist for new configurations. You'll want to add your own procedures and ideas to it.

- Keep a notebook describing the layout of the system, including a history of any changes you or a field support person have made. In particular, you should save hard copy records of your disk label(s).
- Because you are not experienced with the Sun environment, use the defaults provided by SunInstall, `format`, and other files and programs when in doubt.
- Customize your system kernel after you finish running SunInstall.
- Customize the remainder of your environment (and, if applicable, your users environments) only after you've gained experience and some expertise.
- Make backup tapes regularly, including right after you reconfigure your kernel after installation. This may be the system administrator's most crucial task. Without backup tapes, lost files are gone forever.
- If you are going to make a major change to a file system, do full backups first. Do this in addition to the incremental backups that you should perform regularly.
- Plan any changes completely before implementing them. If you forget a step or do something out of order, you might introduce big problems.

- If you run into trouble and are not sure what to do, call Sun Technical Support. Note that if the warranty period has expired on your system, you will be charged for the call unless you have a support contract.
- If you do have to call Sun Technical Support, gather as much information about your problem as is possible before calling. For example, write down information such as error messages, type of configuration you have, and commands that may have caused the problems. Have these on hand when you call.

Introducing the SunOS Operating System

This chapter presents basic information about the SunOS Operating System that you must know to administer your individual machine or network. Topics covered include:

- Essential operating system concepts
- The superuser user name and how to run with `root` privileges
- The *SunOS Reference Manual* and how to use it
- Contents of the SunOS file systems
- Important system and network administration databases

2.1. Introducing the SunOS and UNIX Operating Systems

The SunOS Release 4.1 operating system is the large set of programs and files that control operations for Sun computers. Release 4.1 is based on the UNIX System V operating system developed by AT&T and the 4.3 BSD UNIX operating system developed by the University of California at Berkeley.

Before learning the concepts and procedures in this book, you need a basic knowledge of the SunOS operating system. If you are unfamiliar with the UNIX and SunOS operating systems, these books in your documentation set can help you attain SunOS skills.

SunOS User's Guide: Getting Started

SunOS User's Guide: Doing More

Then refer to this manual for system administration-specific issues, and the *SunOS Reference Manual* for information about individual programs. In addition, you can learn UNIX concepts in any degree of detail from commercially available books on the subject.

As is the case with all UNIX-based operating systems, the SunOS operating system consists of a *kernel* and many hundreds of other files containing data and programs. The kernel controls all the basic functions of the computer, including handling of such devices as disks and printers, as well as scheduling, file management, and network services.

How Does Your Machine Get Its Kernel?

All Sun computers receive their kernels when they are powered up, or *booted*. When a machine with a disk boots, it reads its kernel from the local disk. When you install a new version of the operating system from tape, the kernel is copied to that disk into the single file called `vmunix`. Diskless machines receive their kernels from the NFS file server from which they were configured during SunInstall™.

During booting, the kernel is loaded into memory, where it resides until you shut down the machine. The SunOS operating system loads other files into memory as needed. Though the kernel file is by default called `vmunix`, it is possible to have kernels with other names.

What Does the Kernel Do?

The kernel manages all physical resources of a Sun computer. Some kernel functions are listed below.

- Implementing the *file system*, the tree-structured hierarchy of directories and files on the local disk or accessed remotely from a server
- Permitting processes to create, read, write, and access these files

A *process* is an operating program. For example, you use the `cp` command to copy a file. While `cp` is running, it is considered a process.

- Managing *daemons*

Daemons are processes that start during booting and should run constantly until you halt the machine. Daemons perform major system functions, such as mounting file systems and routing electronic mail. For example, the line printer daemon (`lpd`) handles print requests. If users intend to print, their machines must run `lpd`, whether they use remote printers on their network or printers attached to their machines. If `lpd` stops running, you have to restart it before users can print.

- Moving large groups of information called *pages* from the disk into main memory, as needed. This concept is known as *virtual memory*.
- Keeping track of all active processes and deciding which gets to run next
- Managing *device drivers*

These are programs that control physical devices, such as the workstation screen, graphics display, mouse, keyboard, disk, tape, RS-232 serial ports, and Ethernet IC boards.

- Managing the networking software that implements network services
- Managing interprocess communication facilities such as sockets, which are involved in network activities
- Providing facilities for creating, examining, and modifying processes
- Providing system management functions, such as booting, halting, and error handling.
- Providing miscellaneous functions that make system resources (memory, timers, and the like) available to processes.

What's in the Rest of the Operating System

Hundreds of other files, programs, and shell scripts comprise the operating system. Most are *commands*, also called *utilities*, which you invoke directly at your shell's command line. For example, `cd` and `ls` are two of the most essential commands.

Other files contain *libraries*. These are collections of software routines, which programmers select and incorporate into their programs. Libraries form an extension of the basic system features implemented by the kernel. Network services such as NIS use libraries extensively.

Many operating system files contain data used by software applications and user programs. Moreover, files in the directory `/etc` contain information that a machine needs to know in order to perform basic functions. These files are forms of *system administration databases*, which maintain information that SunOS programs read while performing their associated tasks. The last section in this chapter introduces these databases.

2.2. Becoming Superuser

The SunOS operating system provides the special user name `root` for use in system administration. When you log in with this name, you become the most privileged user on the system, the *superuser*. As superuser, you have permission to run critical system administration programs and edit sensitive files denied to regular users. Some tasks that require superuser privileges include these.

- Editing sensitive files like the kernel configuration file or the password file.
- Changing permissions for files and directories other than those you personally own.
- Running programs that enable you to add new users, groups, and equipment.

You can become superuser in either of two ways.

- By logging in as `root` in response to the login prompt.
- By typing `su` from a shell where you are logged in under your regular user name.

You will see the expressions “superuser” and “root” (implying the root user name) used interchangeably throughout SunOS documentation. For example, some instructions in this manual will tell you to become superuser, while others will tell you to log in as “root” and do something. This simply means enter either `su` at command level (`%` or `$` shells) or `root` at the login screen to become superuser.

Assigning a Password to the Root User Name

Because the superuser is essentially the overlord of the machine, it is imperative for the root user name to have a password. If your machine was already in use when you became administrator, ask the previous administrator for its root password. Then you can change the password to one you can remember. However, if your machine is brand new, it will not have a password assigned to root.

The following instructions show how to log in as root from a screen displaying the login prompt, then add or change the root password. Note that you will find yourself in this situation right after booting up a brand new Sun computer.

1. Enter the following to log in as superuser.

```
Login: root
Password:
```

2. If you are logging in to a brand new system, you will automatically be logged in as root. If your system is already in use, type the password its previous administrator gave to you. The system then displays a pound sign (#) prompt, signifying that you are now logged in as root.
3. Change the password as shown below.

```
# passwd root
# New password: your password
# Retype new password: your password
```

where *your password* is the password you want for the root user name. The password should be at least five characters long; do not use your name or another obvious word as the root password. The system asks you to type the new password twice. If you do not type the same letters in response to the retype prompt, the system will not let you change the password. Therefore, if you make a typing error, you won't wind up with a password you do not know.

Logging in as Superuser from a Shell

For safety's sake, do not log in as root to perform any operations other than system administrative ones. Therefore, develop the habit of logging in with your personal user name, becoming superuser for system administration tasks, then returning to your regular shell. To do this, you use the `su` command from the shell as follows.

```
% su
Password: root password
# operations you want to perform
# exit
%
```

You can type the `exit` command or press **Control-D** to return to your regular shell.

2.3. Using the SunOS Reference Manual

The *SunOS Reference Manual* provided in your document set is a comprehensive reference tool, describing all commands, system programs, and system files included in the SunOS operating system. It is divided into eight sections with alphabetically listed entries, much like an encyclopedia. Each command, program, or file has an individual entry, referred to as a manual page, or, colloquially, as a *man page*.

The sections contain the following information.

- Section 1 describes the user commands, sometimes referred to as utilities or applications.

- Section 2 describes system calls.
- Section 3 describes library routines.
- Section 4 describes the special files used for devices.
- Section 5 explains major system files, many of which are used for system administration.
- Section 6 describes games and demonstration programs available from Sun.
- Section 7 describes what are known as “public files.” These files include various tables and macro packages used with the `nroff` and `troff` text formatting programs.
- Section 8 describes the system administration and maintenance commands and daemons.

Throughout this manual, you will see many references to the entries in the *SunOS Reference Manual*. Each reference contains the name of the command, plus the section where you can find its descriptive man page. For example, in this *System and Network Administration* manual, a reference to the man page for the `/etc/passwd` file would look like the following.

```
passwd(5)
```

This means that the entry is in Section 5 of the *SunOS Reference Manual*, and its title is `passwd(5)`. By contrast, a reference to the `passwd` command would look like `passwd(1)`, since this entry is in Section 1 of the *SunOS Reference Manual*.

During SunInstall, Release 4.1 allows you to select the online version of the man pages as an option. Should you choose this option, you can display a formatted version of a particular man page on your screen. The man pages are located in the directory `/usr/share/man`. Subdirectories either begin with the word `manx` or `catx` where `x` is a number from 1 to 8, indicating a section in the *SunOS Reference Manual*. The `man1-8` directories contain all man pages in `troff` source format. The `cat1-8` directories contain all man pages as formatted output.

Note: If you decide to use online man pages, you also must select the text processing option during SunInstall

Displaying an Online Man Page

To display a formatted man page on your screen, use the `man` command as follows.

```
% man entry_name
```

where `entry_name` is the name of the entry you want to display.

Some man pages have the same name but are in separate sections. The classic example is `passwd`, discussed previously. If you know which section the man page you want is in, you can specify its section in the `man` command.

```
% man 5 passwd
```

As a result, you receive online man page `passwd(5)`, describing the `passwd`

database (/etc/passwd file). If you do not specify a section

```
% man passwd
```

you receive the man page entry passwd(1), describing the passwd command.

If you use the /usr/etc/catman command to create the whatis database, you can find out what other man pages might give more information about a particular command. To build this database, enter the following.

```
% /usr/etc/catman -w
```

(Refer to the catman(8) and whatis(1) man pages for complete details.) Thereafter, you can obtain information for particular commands by using man with the -k option.

As an example, you could enter this.

```
% man -k lpr
```

Your screen would display the following:

```
lpr (1)           - send a job to the printer
lprm (1)          - remove jobs from the printer queue
```

You receive information not only about the lpr command, but also the associated lprm command.

2.4. The SunOS File Systems

As you discovered when reading *Installing SunOS 4.1* and the user guides, the SunOS operating system is organized as a series of hierarchical file systems. This next section describes the file systems that are included in Release 4.1, and their important programs and files.

How Your Machine Gets its File Systems

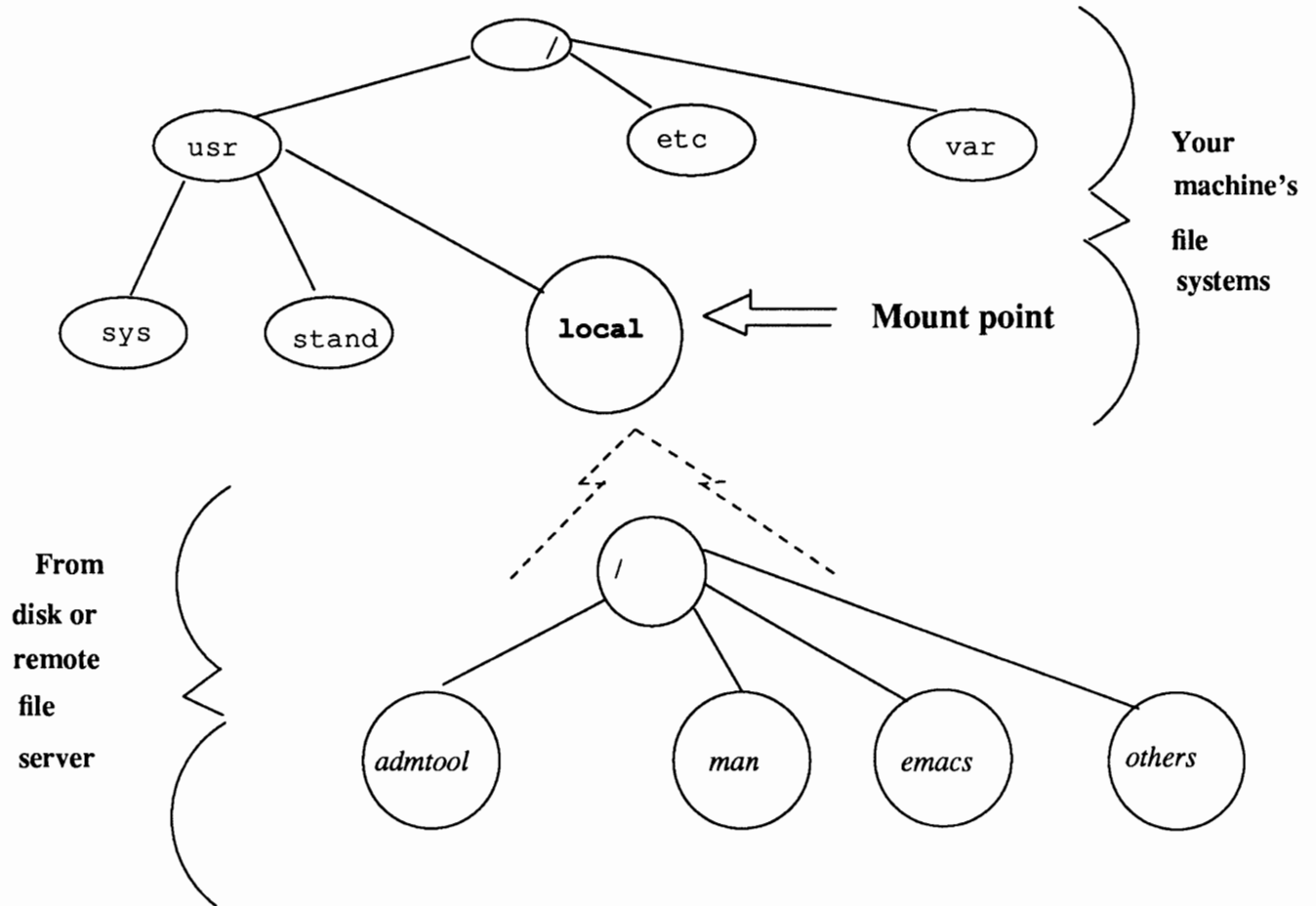
Machines obtain their file systems by *mounting* them. In a traditional UNIX environment, mounting is the process of accessing a file system, or a directory hierarchy of a file system, from a disk or tape. In a networked SunOS environment, computers also access file systems over the network from file servers. SunOS Release 4.1 offers additional types of mounts, which enable you to create new types of file systems. You'll find a brief description of the new file system types at the end of this section.

When your machine boots up, it automatically runs the SunOS mount program and mounts the files it needs for operation, either from the network or its local disk. You also can run the mount program by hand to access additional files, such as application software and user programs.

When a machine mounts a directory hierarchy, it attaches the directory to a *mount point*. Mount points are locations within your machine's local file systems, through which your machine accesses the directories it mounts. Usually, a mount point is an empty directory.

Figure 2-1 illustrates the mounting process.

Figure 2-1 *Mounting the /usr/local Directory*



This figure shows how a directory hierarchy is mounted on the mount point `/usr/local`. When you install Release 4.1 on a server, `/usr/local` is an empty directory. Figure 2-1 2-1 shows how the directory hierarchy `/usr/local`, obtained from a local device or over the network, is mounted on your formerly empty `/usr/local` mount point. Thereafter, users on your machine can go to `/usr/local` and run the many programs now installed in that location.

The traditional UNIX mount—mounting files from a locally attached disk or tape device—is often called a *4.2 mount* in this manual. The chapter *File Maintenance* describes these local mounts in detail. A machine with a disk mounts files on the disk using 4.2 mounts and uses other types of mounts to obtain files over the network. If you have a non-networked standalone or time-sharing system, your machine can only perform 4.2 mounts.

Networked machines can mount files from a remote file server. SunOS networks provide software for doing two types of remote mounts: NFS and RFS. The type of mount your machine must do to access remote files depends on the software the file server uses to share them.

All Sun computers use NFS file sharing software by default. When an NFS file server announces that its files are available to others, it is said to *export* them. Your machine must do an *NFS mount* to access these files. The chapters *The Sun Network File System Service* and *Using the NFS Automounter* fully describe how to use NFS.

In addition, some Sun file servers may share their files using RFS software. When an RFS file server announces that its files are available to others, it is said to *advertise* them. If you mount a file system from an RFS file server, your machine must perform an *RFS mount*. The chapter *The Remote File Sharing Service* discusses RFS in detail.

File Systems on a Client

A client machine must have the SunOS file systems `/` (root) and `/usr` in order to operate.

- Standalones have `/` in Partition a and `/usr` in Partition g, unless you specify otherwise. SunInstall also configures Partition h as `/home` unless you specify otherwise, or the disk's capacity is less than 110 MB.
- Dataless clients have `/` installed on their local disks but must mount `/usr` from an NFS file server.
- Diskless clients receive `/`, `/usr`, and `/home` when they boot from their NFS file servers.

You can run SunInstall and configure a client with a disk to receive `/` and/or `/usr` from an NFS file server. Additionally, you can configure a client to receive its `/home` directory from an NFS or RFS file server.

File Systems on a File Server

The primary function of a file server is to share its files with other machines. Nevertheless, file servers require their own local `/` and `/usr` file systems. SunInstall respectively loads them into Partitions a and g of the first disk on the server, *device_abbreviation0*. (These abbreviations are: `xd0` for the first Xylogics 7053 disk, `xy0` for a Xylogics 450 or 451, or `sd0` for a SCSI disk.)

Additionally, an NFS file server by default provides four other file systems that its clients can access over the network.

```
/home
/export/root
/export/swap
/export/exec
```

These file systems are installed initially on the NFS server during SunInstall. The clients then access these files while booting.

Swap Space

All machines must have a reserved part of the disk for the kernel to use during processing. This area is called *swap space*. SunInstall reserves Partition b as swap space on all machines with disks, including servers. You can change this during SunInstall or enlarge (but not decrease) swap space, as described later in the chapter *Reconfiguring the System Kernel*. Diskless clients access a swap file system from their NFS servers called `/export/swap/client_name`.

Importance of Symbolic Links

The SunOS file system, as with all UNIX- based file systems, is organized as a tree-structured hierarchy of directories, device nodes, symbolic links, and ordinary files. Since Release 4.0, SunOS file systems are organized somewhat differently than the traditional 4.2 BSD file organization. These files and devices could reside anywhere in the hierarchy. However, previously-existing commands expect certain files and devices to reside in specific directories, and will not function properly unless they do. Because programs expect to find files in their traditional directories, in SunOS Release 4.1 these directories contain *symbolic links*, rather than the actual files.

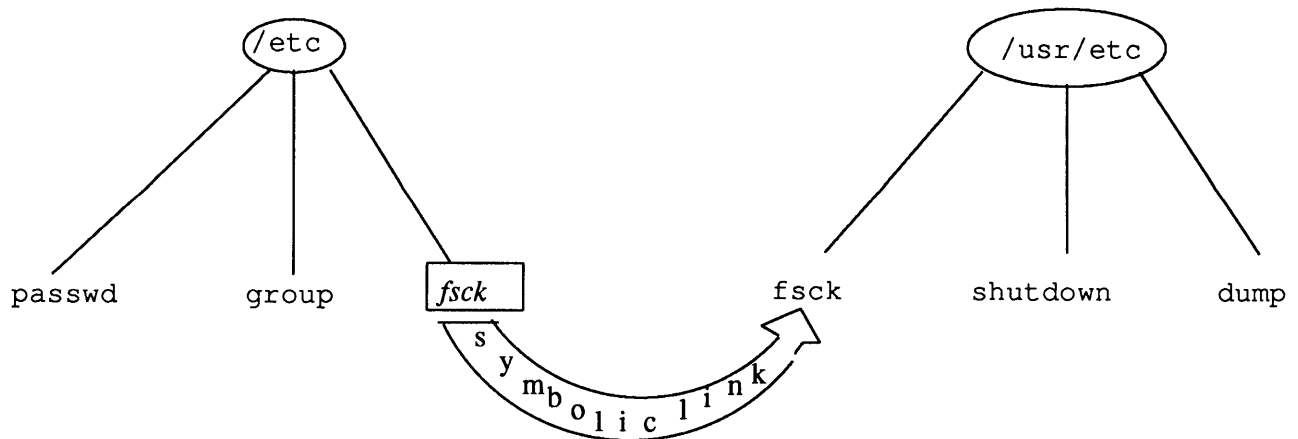
A symbolic link is a file entry that points to another file in a different location. For example, commands such as `chown` and `fsck` have been moved from their standard locations in `/etc` to the directory `/usr/etc`. In Release 4.1, if you ask for a long listing of the `/etc` directory, you find the following entry for the `fsck` (file system check) command:

```
lrwxrwxrwx 1 root 13 Jun 4 18:27 fsck -> /usr/etc/fsck
```

The `l` preceding the list of permissions (`rw-rw-rw`) indicates that this is a symbolic link, not an actual file. The phrase `fsck -> /usr/etc/fsck` indicates that the `fsck` command file is actually in the directory `/usr/etc`.

The next illustration shows how symbolic links work. Note how the file `/etc/fsck` is really a symbolic link to the actual `fsck` program, which resides in `/usr/etc/fsck`.

Figure 2-2 A Symbolic Link



Symbolic links make it unnecessary for you (or a program) to learn the locations of moved files. You can type the old pathname or the new pathname for a symbolically-linked command. It will execute when specified either way.

Suppose you type `/etc/fsck`, as you would in prior Sun releases or other versions of UNIX and the BSD UNIX operating system. This causes the kernel to look in `/etc` for `fsck` and find the symbolic link, rather than the actual file. The kernel reads the link, goes to the file indicated by the link (`/usr/etc/fsck`), and executes the command.

The root File System

The root file system (/) is the first required SunOS file system, located at the top of the hierarchical file system tree. It contains the files and directories crucial for system operation, such as the kernel binary image, a device directory, and programs used for booting. The root directory also contains mount points where file systems mounted by SunInstall (and you) can connect to the root file system hierarchy.

The root file system contains the following files and directories.

bin@	home/	net/	usr/
boot	kadb*	sbin/	var/
dev/	lib@	sys@	vmunix*
etc/	lost+found/	tmp/	
export/	mnt/	tftpboot/	

The symbols following file names indicate that: /, the file is a directory; @, a symbolic link; *, an executable file. The files in / are described as follows.

- `/bin` In traditional UNIX file systems, this directory contains user commands. However, in all releases from 4.0 onward, the `/bin` directory is actually a symbolic link to `/usr/bin`, which is described in the section, *The `usr` File System*.
- `/boot` On machines with a disk, this file contains a program for booting the system.
- `/dev` This is the “device” directory, which contains all of the *device special files*, also known as *device nodes*. Among the devices listed are tape drives (for example, `mt0`), printers (such as `pp0`), disk partitions (for example, `sd0a`), and terminals (such as `ttya`). Note that the contents of `/dev` differ for each configuration, depending on its equipment. `/dev` also contains a shell script called `MAKEDEV`. `MAKEDEV` creates device nodes for all devices supported by the SunOS operating system that you add to your configuration, as explained in the chapter *Adding Hardware to Your System*.
- `/etc` This “et cetera” directory contains data files and subdirectories used in system administration. Because of its importance, `/etc` and its companion directory `/usr/etc` are described fully in a later section, *The `/etc` and `/usr/etc` Directories*.
- `/export` This directory contains the file hierarchies that clients access with read/write permission from a server. (In other words, clients have permission to edit files in the directories under `/export`.) The `/export` directory hierarchies have major significance to system administration and are discussed in more detail shortly.
- `/home` This is the mount point for users’ home directory hierarchies. You should create personal directories for users under the `/home/server_name` directory. On machines with small disks, the home directories for users reside in `/usr/home`.

<code>/kadb</code>	This is the kernel debugger program.
<code>/lib</code>	This is a symbolic link to the directory <code>/usr/lib</code> , described in the section, <i>The <code>usr</code> File System</i> .
<code>/lost+found</code>	Normally this directory is empty. However, if a file system becomes damaged, the <code>fsck</code> program puts in <code>lost+found</code> links to any files that it cannot link elsewhere in the file system in an acceptable fashion.
<code>/mnt</code>	This is an extra mount point that you can use to temporarily or permanently mount a file system.
<code>/sbin</code>	This directory contains the executable files necessary for bringing up the <code>/usr</code> file system at boot time: <code>hostname</code> , <code>ifconfig</code> , <code>init</code> , <code>mount</code> , and <code>sh</code> , before the <code>/usr</code> file system is mounted.
<code>/sys</code>	This is a symbolic link to the directory <code>/usr/kvm/sys</code> .
<code>/tmp</code>	This is a directory for holding temporary working files. Various SunOS utilities, such as <code>cc</code> , the C compiler, and <code>ar</code> , create temporary data files in <code>/tmp</code> . Users may also use <code>/tmp</code> as a temporary workspace. Every time the system is rebooted, the script <code>/etc/rc</code> removes all files in <code>/tmp</code> except for subdirectories.
<code>/tftpboot</code>	This directory, only present on NFS client servers, contains files used to boot diskless clients over the network.
<code>/usr</code>	This is the mount point for the <code>usr</code> file system. <code>/usr</code> is fully described in its own section.
<code>/var</code>	This important directory is a feature of the SunOS system. The next subsection describes the contents of <code>/var</code> .
<code>/vmmunix</code>	This is the SunOS kernel.

The `/var` Directory Hierarchy

This directory contains subdirectories with data files that tend to grow. If you expect these files to grow quite large, it is a good idea to run SunInstall and assign `/var` to its own partition on the disk. The Release 4.1 `/var` directory hierarchy contains the following directories and files.

<code>adm/</code>	<code>log/</code>	<code>preserve/</code>	<code>tmp/</code>
<code>crash/</code>	<code>net/</code>	<code>spool/</code>	<code>yp/</code>

The major directories in `/var` are described as follows:

<code>/var/adm</code>	This directory holds system accounting files, including those that store console messages.
<code>/var/spool</code>	This directory contains files being processed for printing and electronic mail, among many others. Spool directories hold files pending further processing, typically by a program other than the one that placed the file into the spool directory. For example,

`/var/spool/lpd` holds files sent by the `lpr` command that are waiting to be output to the printer. `/var/spool/mail` contains users' *system mailboxes*, which contain incoming mail waiting to be read. `/var/spool/mail` can be NFS mounted from a mailbox server. `/var/spool/uucp` contains data files used by the `uucp` communications software and work files in transit to or from other machines.

`/var/crash` This directory holds an image of the contents of memory when there is a crash.

`/var/preserve`
This is a directory that holds files saved by the `vi` and `ex` editors if the system should crash.

`/var/yp` This directory contains programs and files that you run to implement `yp` on your system.

`/var/tmp` This directory is similar to `tmp` in the root directory. However, `/var/tmp` is available for users as temporary work space. This directory replaces `/usr/tmp`.

Note that in Release 4.1, the `vi` and `ex` editors keep their temporary buffers in `/var/tmp` rather than `/tmp`. *Unlike /tmp, files are not cleared out of /var/tmp when you reboot.*

The `usr` File System

`/usr` is the second required file system in Release 4.1. It differs from a traditional UNIX `/usr` file system in the following ways.

- Executables traditionally in `/etc` are now in `/usr/etc`.
- Executables traditionally in `/bin` are now in `/usr/bin`.
- Executables traditionally in `/lib` are now in `/usr/lib`.
- Architecture-dependent executables are now in the `/usr/kvm` directory.

Many files in the `/usr` file system are executable commands, system programs, and library routines. These files are completely described in the *SunOS Reference Manual*.

These are the directories in the `/usr` file system.

<code>5bin/</code>	<code>dict/</code>	<code>lib/</code>	<code>nserve@</code>	<code>stand@</code>
<code>5include/</code>	<code>etc/</code>	<code>local/</code>	<code>old/</code>	<code>sys@</code>
<code>5lib/</code>	<code>games/</code>	<code>lost+found/</code>	<code>pub/</code>	<code>tmp@</code>
<code>adm@</code>	<code>hosts/</code>	<code>man@</code>	<code>share/</code>	<code>uch/</code>
<code>bin/</code>	<code>include/</code>	<code>mdec@</code>	<code>spool@</code>	<code>xpg2bin/</code>
<code>boot@</code>	<code>kvm/</code>	<code>net@</code>	<code>src@</code>	<code>xpg2include/</code>
				<code>xpg2lib/</code>

The major directories are described below.

`/usr/bin` This major directory contains the basic SunOS commands you use every day. In Release 4.1, `/usr/bin` contains the

commands traditionally found in `/bin`, such as `ls`, `cat`, and `mkdir`. (`/bin` in the root file system is now a symbolic link to `/usr/bin`.) It also contains basic system administration commands, such as `df` and `chmod`.

System V Directories

The directories `/usr/5bin`, `/usr/5lib`, and `/usr/5include` contain UNIX System V software that cannot be converged with the rest of the release.

Note: The `/usr` file system will contain `5bin`, `5include`, and `5lib` only if you load them as an optional software category while running `SunInstall`

`/usr/5bin` contains UNIX System V programs that are incompatible with those in Berkeley UNIX. For example, the commands `/usr/bin/pr` and `/usr/5bin/pr` have different options. If you want to use System V programs by preference, simply include `/usr/5bin` early in your path. Libraries and include files for compiling System V software reside in `/usr/5lib` and `/usr/5include`, respectively.

System V programs that are upward compatible with those in Berkeley 4.2 UNIX are already in the regular system directories. For example, the new, improved Bourne shell for System V is `/usr/bin/sh` and `/usr/bin/make` has all the System V enhancements.

Programs that existed only on System V have been added to regular system directories as well. For example, the text manipulation programs `cut` and `paste` both reside in `/usr/bin`.

The directories that constitute the System V compatibility package are optional. For optional System V compatibility commands, refer to *System V Enhancements Overview. Installing SunOS 4.1*.

- `/usr/dict` This directory contains English language spelling lists used by the `spell` spelling checker.
- `/usr/etc` This directory contains commands used for system administration and maintenance. It is described more fully in the section, *The /etc and /usr/etc Directories*.
- `/usr/games` This directory contains games. See Section 6 of the *SunOS Reference Manual* and *Installing SunOS 4.1* for instructions on installing games from the distribution tapes.
- `/usr/hosts` This directory contains a script called `MAKEHOSTS`, which creates a symbolic link to the `rsh` command for each host in the `/etc/hosts` file. (`/etc/hosts` is introduced later in this chapter.)
- `/usr/include` This directory contains all the standard “include” files (or “header” files) used in C programs. Individual `.h` files are explained in Sections 2, 3, and 5 of the *SunOS Reference Manual*.

- `/usr/kvm` This directory contains the SunOS kernel binaries. The next subsection, “`/usr/kvm` and Kernel Architectures,” explains this directory fully.
- `/usr/lib` This directory contains the files originally in `/lib`, now a symbolic link. `/usr/lib` is a catch-all for SunOS utility files. These include libraries supporting system functions, programs used by various system utilities (for example, `lint`, `lex`, `spell`), macros for the `troff` text processor, line printer filters, and more.
- The important files in `/usr/lib`, from a system administrator’s perspective, are the `sendmail` files, which are used in the electronic mail system described in the chapters *Administering Electronic Mail* and *Customizing sendmail Configuration Files*.
- `/usr/local` This directory is empty when you first install Release 4.1. You can use it for commands, programs, or other files, such as those you might develop at your site or obtain from third parties (like the Sun User Group) and add after installation. To access these files, the users on your system should add `/usr/local` (and/or `/usr/local/bin`) to their `PATH` environment variables.
- `/usr/lost+found`
Every file system has a `lost+found` directory, which serves the same purpose as `lost+found` in the root file system.
- `/usr/pub` The `pub` directory contains files used in printing.
- `/usr/sccs` This directory contains commands used by `sccs`, the Source Code Control System. See `sccs(1)` and the SCCS tutorial in *Programming Utilities and Libraries* for more information.
- `/usr/share` This directory contains files that can be shared across all architectures. If you loaded them, the online man pages will reside in the directory `/usr/share/man`. In Release 4.0, the subdirectory `/usr/share/sys` held kernel object modules. Now `/usr/share/sys` is a symbolic link to `/usr/kvm/sys`. The subdirectory `/usr/share/lib` holds the `termcap` file and other files pertaining to terminals; it also holds macro packages used by the `troff` command. `/usr/share/src/sun/suntool` has files used by `suntools`.
- `/usr/src` This directory is not present on all configurations. If your machine is licensed to contain source code, it will reside in `/usr/src`.
- `/usr/stand` This directory is a symbolic link to `/usr/kvm/stand`.
- `/usr/sys` This directory is a symbolic link to `/usr/kvm/sys`.

`/usr/ucb` This directory contains commands that are part of Berkeley UNIX. (`ucb` is an abbreviation for University of California at Berkeley.) These include basic user commands, such as `lpr`, `more`, and the `vi` text editor; additionally `/usr/ucb` also contains system administration commands like `finger`, `netstat`, and `tftp`, which are described in Part Three of this manual.

The `/usr/kvm` Directory and Kernel Architectures

The `/usr` file system for Release 4.1 has a new directory hierarchy, `/usr/kvm`. The `kvm` hierarchy reflects the addition of new kernel architectures to support new Sun machines. It is important for you to understand kernel architecture concepts, because they are involved in these functions.

- Reconfiguring kernels for your machine(s), as explained in the chapter *Reconfiguring the System Kernel*
- Adding machines of different architectures to your NFS file server's network, as explained in the chapter *Administering Workstations*.

Each Sun machine has an application architecture and a kernel architecture. The chapter *The System Administrator's Role* introduced the concept of kernel architectures.

A Sun computer's *application architecture* defines the major product family it belongs to and the type of application software it can run. Most importantly, application architecture determines the type of executable programs (programs in `/usr`) that the computer can run. Release 4.1 runs on any machine with the application architecture `sun3` or `sun4`. A machine can run all user programs and software application packages designed for its application architecture, provided that these programs do not depend on kernel structures.

All machines in the Sun-3 product line have the `sun3` application architecture. Thus they can share the files in the `/usr` file system exported by any model Sun-3 server, with the possible exception of the `/usr/kvm` hierarchy, as explained shortly. All machines in the Sun-4 product line have the same application architecture, `sun4`. Therefore, they can share the `/usr` file system exported by any model Sun-4 server or SPARCserver, again with the possible exception of the `/usr/kvm` hierarchy. However, a machine with a `sun3` architecture cannot run the executables mounted from `/usr` on a Sun-4 machine or from a Sun-4 release tape, and vice versa.

The `/usr/kvm` directory hierarchy contains the binary files that make up the machine's actual kernel and are therefore considered *kernel dependent*. The `/usr/kvm` directories differ for each of the four kernel architectures supported by Sun.

```
sun3
sun3x
sun4
sun4c
```

The combination of application architecture and kernel architecture are referred to as an *architecture pair*.

For example, a Sun-3/60 and Sun-3/80 have the same application architecture, `sun3`, but their kernel architectures differ. The Sun-3/60 has the kernel architecture `sun3` (Motorola 68020 chip) and the Sun-3/80 has the `sun3x` kernel architecture (Motorola 68030 chip). These machines can share the files in `/usr`, with the exception of the kernel specific executables in `/usr/kvm`.

Here is a typical `/usr/kvm` directory.

```
boot/          m68k@        ps*          sun386@      u3b2@
crash*        machine@     pstat*      sun3x@       u3b5@
eeprom*      mc68010@    sparc@      sun4@        vax@
i386@        mc68020@    stand/      sys/         vmstat*
iAPX286@     mdec/       sun@        u370@
libkvm.a     modload*    sun2@       u3b@
libkvm.so.0.2* pdp11@     sun3@       u3b15@
```

Many of the files above are symbolic links that give machine identity to the commands by the same name in `/usr/bin`. These files are:

```
i386          sparc        u3b
iAPX286      sun2         u3b15
m68k         sun3         u3b2
mc68010     sun3x        u3b5
mc68020     sun4
pdp11       u370
```

The commands `ps`, `pstat`, and `vmstat` display system statistics. You will learn about them later in this manual.

Note that `machine` is a symbolic link to `/usr/include/machine`. Also, `libkvm.a` is a system library, and `libkvm.so.0.3` is a shared library. The `ldconfig` command and `ld.so` link editor are used to link shared libraries. Shared libraries are fully discussed in the *Programming Utilities and Libraries*.

The directories `mdec` and `stand` contain executables that the machine uses when booting. In particular, `stand` contains programs that must be run as standalone programs from the PROM monitor, rather than run as SunOS processes. The directory `sys` is the “system” directory, containing all the files necessary to build or reconfigure the kernel.

The `boot` directory contains a copy of the actual kernel, `vmunix` or `vmunix_small`.

The `/usr` file system and its `/usr/kvm` directory hierarchy present the “local” view of the file system. Machines configured as NFS file servers and standalones mount `/usr` from their local disk. However, machines that do not have `/usr` on a local disk must obtain the file system by mounting it from an NFS file server. Users on this type of client (diskless and dataless machines) access the executables through their local `/usr` mount point by typing:

```
% cd /usr
```

However, the executables they run either reside in a directory hierarchy under the server’s `/export/exec` file system, or are accessed by a symbolic link in

```
/export/exec.
```

The export File System

This file system contains directory hierarchies that a machine with a disk can “export” with read/write permission to others on a network. On an NFS file server, there are four the default `/export` directories.

```
/export/root
/export/swap
/export/share
/export/exec
```

On a standalone, these are the default `/export` directories.

```
/export/share
/export/exec
/export/lost+found
```

The directory hierarchies of `/export` are defined below.

The `/export/exec` Directory Hierarchy

`/export/exec` is a very significant directory hierarchy for both NFS file servers and clients. When you configure an NFS file server and its clients, SunInstall automatically creates this directory hierarchy. It contains directories, or symbolic links to directories specific to a client’s application architecture (`/usr`) and kernel architecture (`/usr/kvm`). Whether `/export/exec` contains symbolic links or actual directories depends on whether the NFS file server supports one or more architectures, or one or more release levels.

`/export/exec` on a Homogeneous Server

An NFS file server whose clients have the same application and kernel architecture as the server is called a *homogeneous server*.

If you run `ls -l` on `/export/exec` on a homogeneous server, you receive the following display.

```
drwxr-sr-x 3 root  512 Oct  9 20:20 kvm
drwxr-xr-x 2 root 8192 Oct  9 19:40 lost+found
lrwxrwxrwx 1 root   4 Oct  9 19:42 sun4 -> /usr
lrwxrwxrwx 1 root   4 Oct  9 19:42 sun4.sunos.4.1. -> /usr
```

Though the server mounts its `/usr` executables directly from Partition `g` of the local disk, its clients receive their executables by mounting them through symbolic link `/export/exec/application_arch`. This file is simply a symbolic link to `/usr`.

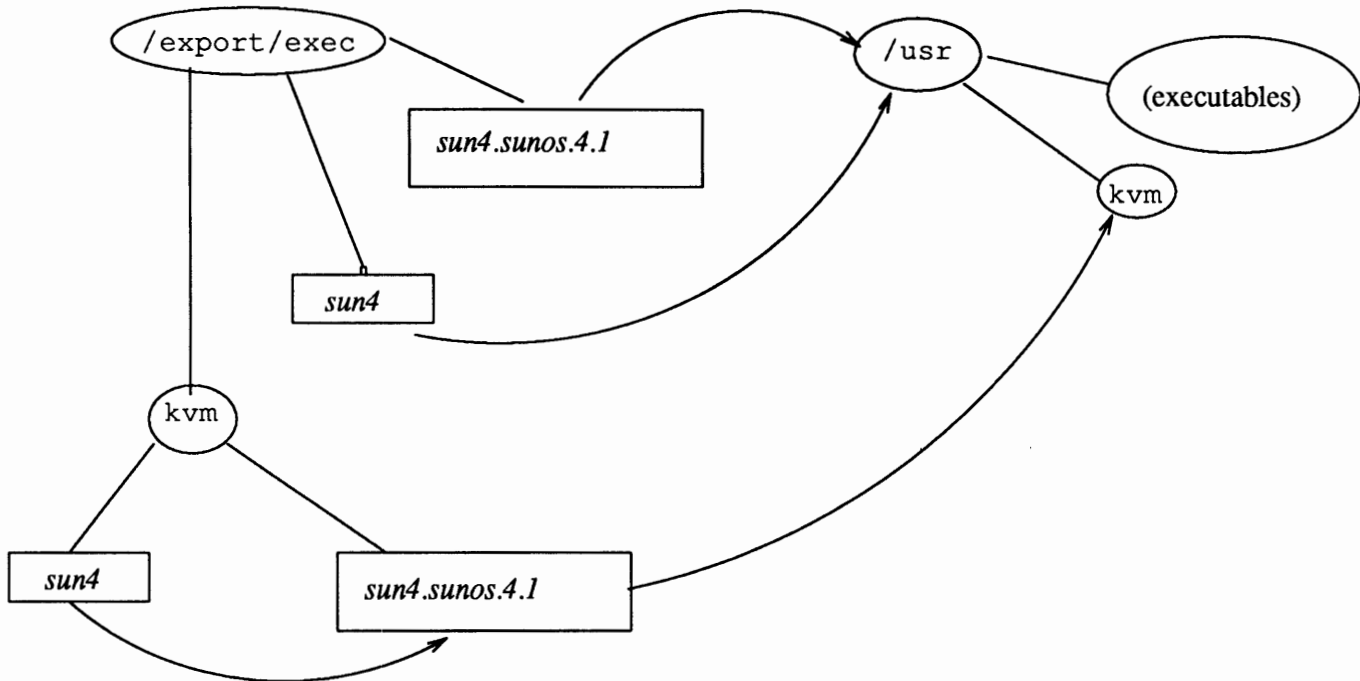
In the listing above, `/export/exec/sun4` provides a symbolic link which the server’s clients, also Sun-4s, use to mount their executable files.

The `/export/exec/kvm` directory on a homogeneous server has the following contents, as shown when you run `ls -l`.

```
lrwxrwxrwx 1 root 23 Oct  9 19:42 sun4 -> sun4.sunos.4.1
lrwxrwxrwx 1 root  8 Oct  9 19:42 sun4.sunos.4.1 -> /usr/kvm
```

Figure 2-3 shows the relationships between files in `/usr`, `/export/exec`, and `/export/exec/kvm`.

Figure 2-3 *Directories with Executables on a Homogeneous Server*



Here the ellipses represent actual directories, and the squares with italicized file names represent symbolic links. The curved lines show how the symbolic links point to directories with real files.

The homogeneous server's `kvm` kernel binaries actually reside in `/usr/kvm`. However, the server's diskless clients must access their kernel files by mounting `/export/exec/kvm/kernel_arch`. In the previous listing, this file is `/export/exec/kvm/sun4`. Note that `/export/exec/kvm/sun4` is really a symbolic link to the release specific kernel architecture file, `/export/exec/kvm/sun4.sunos.4.1`. Because the server supports only SunOS 4.1, `/export/exec/kvm/sun4.sunos.4.1` is a symbolic link to the actual kernel files in `/usr/kvm`.

`/export/exec` on a Heterogeneous Server

An NFS file server that supports clients of more than one architecture type, or supports more than one SunOS release, is called a *heterogeneous server*. The architecture pair of the server is referred to as its *native* architecture. For example, the native architecture of a Sun-4/280 NFS file server consists of `sun4` application architecture and `sun4` kernel architecture.

To support client machines of a different kernel architecture, for example, a SPARCsystem-1 (`sun4c`), you must load an additional `/usr/kvm` directory hierarchy for the `sun4c` architecture on the server. If you want the Sun-4 server to support Sun-3 clients, you must load both the `/usr` and `/usr/kvm` directories for the `sun3` application and kernel architectures. You load this additional

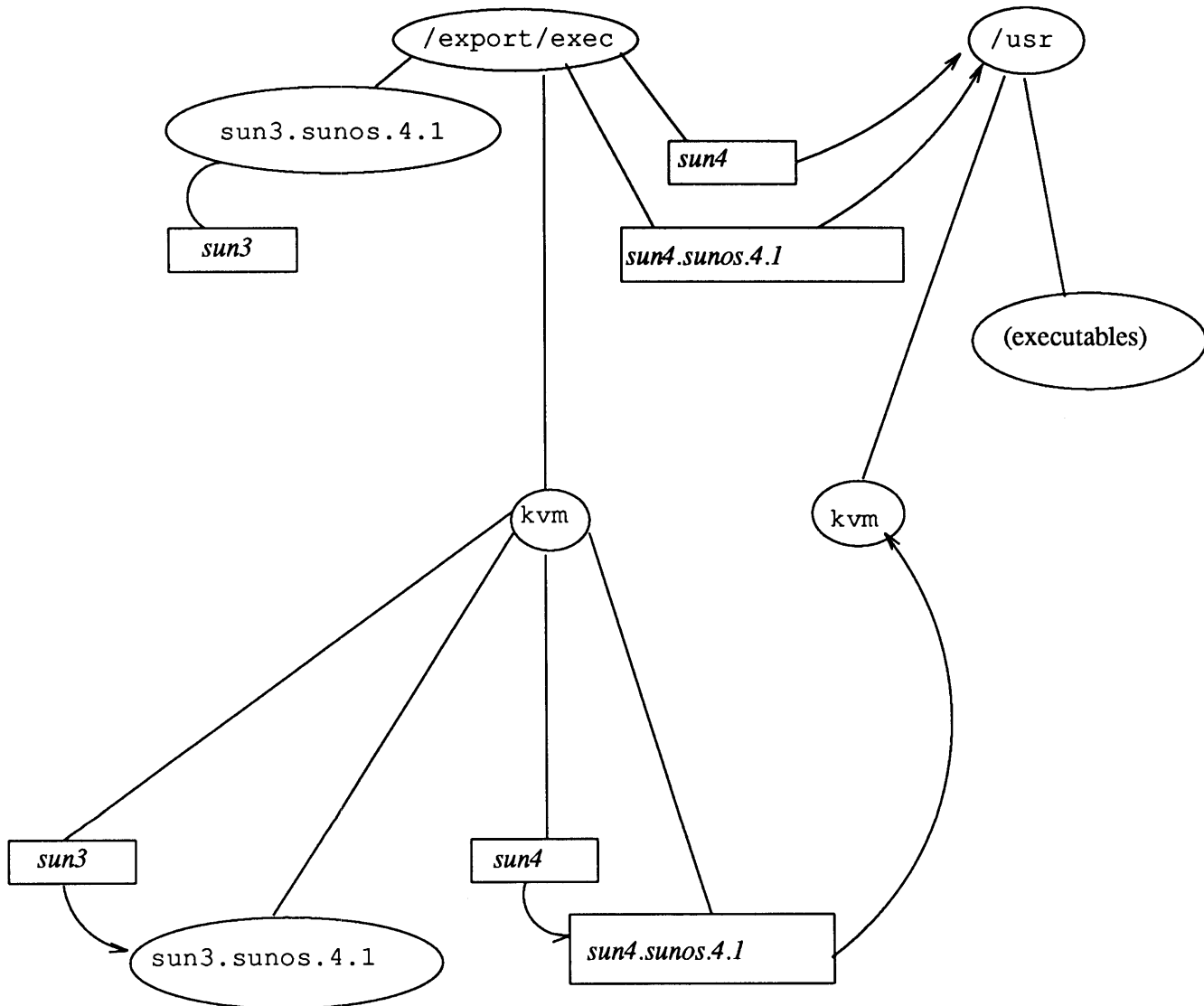
software by running the `add_services` program, as explained in *Administering Workstations*. Once loaded, these additional architectures reside as directories in `/export/exec` and its sub-tree `/export/exec/kvm`.

Here is a directory listing of `/export/exec` for a heterogeneous Sun-4 server.

```
drwxr-sr-x  3 root      512 Oct  9 20:20 kvm
drwxr-xr-x  2 root     8192 Oct  9 19:40 lost+found
lrwxrwxrwx  1 root         23 Oct  9 20:20 sun3 -> sun3.sunos.4.1
drwxr-sr-x 32 root    1024 Oct  9 21:24 sun3.sunos.4.1
lrwxrwxrwx  1 root         4 Oct  9 19:42 sun4 -> /usr
lrwxrwxrwx  1 root         4 Oct  9 19:42 sun4.sunos.4.1 -> /usr
```

Figure 2-4 shows the relationships between directories with real files and symbolic links on a Sun-4 heterogeneous server.

Figure 2-4 Directories with Executables on a Heterogeneous Server



In this figure, rectangles represent symbolic links and ellipses represent directories with real files.

Under this server's `/export/exec` directory, `sun4` and `sun4.sunos.4.1` represent the server's native application architecture. Thus they are both symbolic links to `/usr`. The server's Sun-3 clients use the symbolic link `sun3` to mount their `/usr` executables. But the actual executables are in the `sun3.sunos.4.1` directory, to which `sun3` is merely a symbolic link.

On a heterogeneous server, a listing of the `/export/exec/kvm` directory will resemble the following.

```

lrwxrwxrwx 1 root      23 Oct  9 20:20 sun3 -> sun3.sunos.4.1
drwxr-sr-x 6 root     1024 Oct  9 20:30 sun3.sunos.4.1
lrwxrwxrwx 1 root      23 Oct  9 19:42 sun4 -> sun4.sunos.4.1
lrwxrwxrwx 1 root       8 Oct  9 19:42 sun4.sunos.4.1 -> /usr/kvm

```

Here `sun4` is the symbolic link by which clients of the same native kernel architecture mount their kernel files. `sun4` points to `sun4.sunos.4.1`, the symbolic link to `/usr/kvm`. The server's Sun-3 clients mount their kernel files through the symbolic link `sun3`. Unlike the link `sun4`, `sun3` points to a Sun-3 `kvm` directory hierarchy, installed under `sun3.sunos.4.1`.

Other Directories in `/export`

On an NFS file server, the `/export` directory contains additional directories its diskless clients must mount in order to operate. They are outlined below:

The `/export/root` Directory

This directory contains the root directories for all diskless clients of an NFS server. Each of these root directories has the name `/export/root/client_name`. The client root hierarchy has the same file organization as the NFS file server's root. Administrators of diskless clients can change files in their machines' root directories to affect their local environments.

The `/export/swap` Directory

This directory contains the individual swap areas for each diskless client of an NFS file server. As is the case with client roots, clients mount their individual swap files, `/export/swap/client_name` from the NFS file server when they boot. Together, the `/export/root` and `/export/swap` are collectively referred to as the client's *bootparams*.

The `/export/share/sunos.4.1` Directory

This directory contains a symbolic link to `/usr/share`, explained earlier. Diskless clients mount the files in `/usr/share` through the `/export/share/sunos.4.1` symbolic link.

The home Directory Tree

When you use SunInstall to set up a machine with a disk, it automatically creates the file system `/home` on Partition `h`. If your machine is an NFS file server, set up home directories in this file system for every user the server supports. This can include users on client machines, individuals who log directly into the file server, and any users on terminals connected to the server. After you finish running SunInstall on the NFS server, you have to create these home directories, as explained in mount their home directories from the file: `/home/server_name/`.

If you administer a standalone or dataless client, you can use the `/home` directory as the top directory for your personal files. Alternatively, you can have the administrator of an NFS file server create home directories for you and, if applicable, others who use your machine, within the server's `/home` directory hierarchy. You can then mount these directories after creating mount points in the `/home` directory in your machine's local file system.

If your machine is a non-networked, time-sharing system, create home directories in `/home` for all users on terminals connected to your machine.

The /etc and /usr/etc Directories

The /etc and /usr/etc directories contain most of the files, commands, and subdirectories that you need for system administration. /etc contains the major files that are a form of the local administrative and network databases on your system.

The files in /etc are listed below.

adm@	inetd.conf	restore@
aliases	install/	rmt@
aliases.dir	ld.so.cache	rpc
aliases.pag	link@	rrestore@
arp@	magic	sendmail.cf
bootparams**	mkfs	services
chown@	mknod@	shutdown@
chroot@	motd@	sm/
clri@	mount@	sm.bak/
config@	mount_rfs@	spool@
crash@	mtab	state
cron@	ncheck@	syslog.conf
defaultdomain	netgroup++	syslog.pid
dkinfo@	netid++	termcap@
dmesg@	netmasks	tmp@
dump@	networks	ttys
dumpdates	newfs@	ttytab
ethers**	nserve/	umount@
exports**	passwd	unlink@
fastboot@	printcap	update@
fasthalt@	protocols	utmp
format.dat	psdatabase	uucp/
fsck@	pstat@	vipw@
fstab	publickey	wtmp
fuser@	rc	xtab
gettytab	rc.boot	yplibind.lock
group	rc.local	
halt@	rc.single*	
hostname.ie0	rdump@	
hosts	reboot@	
hosts.equiv	remote	
ifconfig@	renice@	

One asterisk next to a file indicates that it is an executable file. Two asterisks (**) indicates that these administrative files are found only on client servers. Two plus signs (++) indicates files that are only present on the NIS master server. Files followed by the at sign "@" are symbolic links to other locations, principally /usr/etc and /var. The next section, "Major System Administration Databases," introduces many files in /etc. You can also find reference material about them in *Booting Up and Shutting Down Your System* and the *SunOS Reference Manual*.

With traditional UNIX systems, the /etc directory hierarchy contains the commands used for system administration. In /etc for SunOS Release 4.1, symbolic links exist in place of each command. The actual commands now reside in the directory /usr/etc. You use the administrative commands for performing

basic administrative tasks, such as shutting down the system and backing up files. /usr/etc also contains daemon programs that carry out major administrative functions.

The following files are in /usr/etc.

ac*	fuser*	mconnect*	rfsetup*	unlink*
arp*	gencat*	mkfile*	rmt*	update*
audit_warn*	gettable*	mkfs*	route*	upgrade/
auditd*	getty*	mknod*	rpc.bootparamd*	vipw*
automount*	gpconfig*	mkproto*	rpc.etherd*	vmpage*
biod*	grpck*	modload@	rpc.lockd*	yp/
catman*	halt*	modstat*	rpc.mountd*	ypbind*
chill*	htable*	modunload*	rpc.rexd*	ypserv*
chown*	icheck*	mount*	rpc.rquotad*	ypxfrd*
chroot*	ifconfig*	mount_hfs*	rpc.rstatd*	zdump*
chrtbl*	in.comsat*	mount_lo*	rpc.rusersd*	zic*
clri*	in.fingerd*	mount_rfs*	rpc.rwalld*	
colldef*	in.ftpd*	mount_tfs*	rpc.showfhd*	
config*	in.named*	mount_tmp*	rpc.sprayd*	
crash@	in.named-xfer*	ncheck*	rpc.statd*	
cron*	in.rexecd*	ndbootd*	rpc.yppasswdd*	
dbconfig*	in.rlogind*	newfs*	rpc.ypupdated*	
dcheck*	in.routed*	newkey*	rpcinfo*	
devinfo*	in.rshd*	nfsd*	rrestore@	
devnm*	in.rwhod*	nfsstat*	rusage*	
dkctl*	in.talkd*	nserve*	rwall*	
dkinfo*	in.telnetd*	nslookup*	sa*	
dmesg*	in.tftpd*	nstest*	savecore*	
dump*	in.tnamed*	pac*	showfh*	
dumpfs*	inetd*	ping*	showmount*	
edquota*	install/	portmap*	shutdown*	
eeeprom*	installtxt*	pstat@	spray*	
etherfind*	intr*	pwck*	swapon*	
exportfs*	keyenvoy*	quot*	syslogd*	
extract_unbundled*	keyenvoy*	quotacheck*	termcap@	
fastboot*	keyenvoy*	quotaoff*	tfsd*	
fasthalt*	kgmon*	quotaon*	trpt*	
foption*	ldconfig*	rarpd*	ttysoftcar*	
format*	link*	rdump@	tunefs*	
fpuversion4	listen*	reboot*	tvconfig*	
fsck*	lpc*	renice*	tzsetup*	
fsirand*	mailstats*	repquota*	umount*	
		restore*	umount_tfs*	

Six files are symbolic links.

File	New Location
crash	/usr/kvm/crash
modload	/usr/kvm/modload
pstat	/usr/kvm/pstat
rdump	/usr/etc/dump
rrestore	/usr/etc/restore
termcap	/usr/share/lib/termcap

`/usr/etc` primarily contains executable files and a few crucial directories, which are described in greater detail throughout this manual and in the man pages. It does have a few subdirectories, the most significant of which is `yp`. This directory contains commands for setting up and maintaining the NIS service. (NIS is explained in more detail in *Introducing Networks* and in *The Network Information Service*.)

Additional File Systems Supported by Release 4.1

SunOS Release 4.1 enables you to access three other types of file systems in addition to the file system types discussed thus far.

- loopback file system
- High Sierra file system
- `tmpfs` file system

You can access these file systems by issuing the `mount` command with the `-t` option, as described in the `mount(8)` man page.

The loopback file system allows you to create virtual file systems. Once a virtual file system has been mounted, it can be accessed through alternate pathnames. You can mount other file systems on it without affecting the original file system. Refer to the `lofs(4s)` man page for more information.

The High Sierra (`hsfs`) file system is a file system type used on CDROM disks. You can install SunOS Release 4.1 from CDROM disks, which use a superset of the High Sierra file system format. The `hsfs` mount type creates a High Sierra file system on the CDROM disk. Refer to the `mount(8)` and `sr(4)` man pages for more information.

The `tmpfs` file system enables you to use the operating system's virtual memory resources as a file system. The subsection below explains how to create a `tmpfs` file system and the benefits `tmpfs` provides. The text assumes you have some system administration experience, but you needn't be an expert to use the procedures specified.

Setting Up the `tmpfs` Memory-based File System

You use `tmpfs` primarily as a performance enhancement, where short-lived files can be written and accessed without any performance impact associated with writing information to a disk or network. Files in `tmpfs` mounted directories exist as long as the `tmpfs` file system is mounted, and you do not shut down your computer. When you reboot your machine or unmount the `tmpfs` file system, all files under `tmpfs` are lost.

The `/tmp` directory is probably the most common place where short term files are used. It is the recommended mount point for `tmpfs` file systems.

Who Should Use `tmpfs` File Systems?

Using `tmpfs` file systems can improve system performance by saving the cost of reading and writing temporary files to a local disk or across a network. `tmpfs` can speed up a machine that has to perform many local compilations. For example, temporary files are created for use during the compilation process. The operating system must generate a lot of disk or network input and output activity while manipulating these files—which are deleted shortly after their creation.

Most sites can benefit from using `tmpfs`, although the difference in performance gains for your machine may vary, depending on system usage and resources. In general, a machine with a lot of physical memory but a slow disk, or on a busy network, will notice improvements from `tmpfs` much more than a machine with minimal physical memory and a fast local disk. This is because `tmpfs` essentially caches the writes in memory normally scheduled for a disk or network file system. Again, performance gains vary depending on use. (Refer to the `tmpfs(4s)` man page for complete technical details.)

A problem that can arise on a machine with a `tmpfs` file system is running out of swap space. This may occur for two reasons:

- The `tmpfs` file system can actually run out of space, just as traditional file system can run out of space.
- A program cannot execute due to lack of swap space. This is because `tmpfs` allocates system swap space to save file data, if it needs to.

Even if your machine uses `tmpfs` and habitually runs out of swap space, you can still benefit by allocating more swap space. You do this by using the `swapon` command, as described in *Reconfiguring the System Kernel* and the `swapon(8)` man page. Remember that when you allocate additional swap space, it is at the expense of additional disk space.

Suggestions for Using `tmpfs`

`tmpfs` is easy to set up and use. Consider which file hierarchies you might want to mount as `tmpfs` file systems. The `/tmp` directory is recommended, since many SunOS programs use it for their temporary files. Designating this directory as the mount point for a `tmpfs` file system will most noticeably improve your system's performance.

You can have your machine mount multiple `tmpfs` file systems as needed. Nevertheless, be aware that they all share the demands of the same system resource. Files created under one `tmpfs` directory use up the space available for any other `tmpfs` file system.

Because files residing in `tmpfs` directories do not survive across reboots or unmounts, you *must not* mount `tmpfs` file systems under `/var/tmp`. This is because the `vi-r` command expects to find preserved files there after a machine reboots.

Procedures for Setting Up tmpfs

To use tmpfs on your machine, you first must include tmpfs information in the kernel. If you have not yet reconfigured your machine's kernel, you should refer to *Reconfiguring the System Kernel* before continuing these instructions. If you have some experience with kernel configuration procedures, continue as follows:

1. Become superuser on your machine.
2. Follow the instructions in *Reconfiguring the System Kernel* for editing the kernel configuration file, to the point where you determine which entries to pick for your machine.
3. Make sure your kernel has the following options line. You'll find it at the end of the options list in the GENERIC kernel file for all kernel architectures:

```
options TMPFS
```

This entry enables tmpfs. If your machine uses the GENERIC kernel, make sure this entry is not commented out.

4. Complete kernel reconfiguration, as shown in *Reconfiguring the System Kernel*.

You can create and mount tmpfs directories either by issuing a mount command on the command line or placing an entry in the `/etc/fstab` file. The syntax of the mount command for tmpfs is:

```
# mount -t tmp swap directory_name
```

tmp is the file system type that must be specified after the `-t` option. You also need to specify swap as shown to set up tmpfs. *directory_name* is the directory to become a tmpfs file system.

The `/etc/rc.local` file contains commands that mount a tmpfs file system on `/tmp` at boot time, but these commands are commented out by default. To mount and use tmpfs in the recommended manner and have the system automatically mount a tmpfs file system when it boots up, follow these procedures:

1. Become superuser and edit `/etc/rc.local`.
2. Remove the comment (`#`) from the following line:

```
mount /tmp
```

3. Exit the file, then edit `/etc/fstab`.
4. Add the line

```
swap /tmp tmp rw 0 0
```

to enable a tmpfs file system in `/tmp`.

5. Reboot your system, as described in *Booting Up and Shutting Down Your System*.

2.5. Major System Administration Databases and Files

As system administrator, you will constantly update or consult a series of system administration databases and other local files to keep your machine and/or your network running smoothly. Each system administration database is represented by a file in `/etc` and, in most cases, a corresponding distributed network database used by services such as NIS or Domain Name Service (DNS). DNS is described in *Introducing Networks* and in *Administering Domain Name Service*. For example, the `passwd` database has two representations, the `/etc/passwd` file and, if NIS is present on your network, the NIS `passwd` maps.

The system administration databases can be separated into two categories: *local* and *network*. You use local administration databases for managing local operations on all machines, from diskless client to network server. Conversely, the network databases are significant to you only if you are administering a network.

Note: If your network does not use NIS or DNS, or your machine is a non-networked standalone, your system administration databases will only be represented by files in `/etc`.

This next section introduces the major system administration databases as they are represented by the files in `/etc`. (Later chapters in Part Three explain network databases.) The file descriptions are organized according to the configuration type(s) that typically use(s) them. Refer to the section that applies to your configuration for information about the file that you need to learn about.

Files for Administering Diskless Clients

Note: These files are present on all systems, not just diskless clients.

If you are managing a diskless client, you only have to maintain *local* files—those that apply specifically to your machine.

```
/var/spool/cron/crontabs/*
/etc/fstab
/etc/passwd
/etc/hosts.equiv
/etc/group
/etc/aliases
/etc/printcap
```

They are introduced below.

Note: If your client is on a network running NIS, some of your local files will not have all the entries shown in this subsection.

Even if you are the only user on the diskless client, this may not always be the case in the future. If others are going to locally log in to your system, make entries for them in the `/etc/passwd` file. If your site plans to organize users in groups, include others who directly log in to your machine in `/etc/group`. You may also want to create mail aliases for them in `/etc/aliases`.

`crontab`

You use the `crontab -e` command to create files that list commands to be executed at specified times. These files are often referred to collectively as `crontab` files, though each file will have the name of the user that created it. The system `crontab` file is located in `/var/spool/cron/crontabs/root`.

Use `crontab` for scheduling commands, shell scripts, and other programs that you want to run at regular intervals, such as every day, once a week, once an hour, and so forth. The `/usr/etc/cron` daemon executes the list of

commands in the crontab file at the specified times.

Here is a typical crontab file:

```
0 * * * * /usr/lib/acct/ckpacct
0 1 * * 1-6 /usr/lib/acct/dodisk
0 2 * * 1-6 /usr/lib/acct/runacct 2> /usr/adm/acct/nite/fd2log
15 5 1 * * /usr/lib/acct/monacct
7 2 * * * /usr/etc/fpa/fparel >/dev/null 2>&1
15 3 * * * find / -name .nfs -mtime +7 -exec rm -f {} ; -o -fstype nfs -prune
5 4 * * 6 /usr/lib/newsyslog >/dev/null 2>&1
15 4 * * * find /var/preserve/ -mtime +7 -a -exec rm -f {} ;
```

Refer to *Administering Workstations*, ,crontab(5) and cron(8) in the *SunOS Reference Manual* for more information.

fstab

The `/etc/fstab` file shows all file systems that your client machine mounts when it boots. Every time you boot your system, `fstab` is read by commands that mount and unmount file systems, and check file system consistency. The system also reads `/etc/fstab` when providing additional swap space.

Here is an example of a typical client's `fstab` file.

```
dancers:/export/root/samba / nfs rw 0 0
dancers:/export/exec/sun3 /usr nfs ro 0 0
dancers:/export/exec/kvm/sun3x /usr/kvm nfs ro 0 0
dancers:/export/share/sunos.4.1 /usr/share nfs ro 0 0
dancers:/home/dancers /home/dancers nfs rw 0 0
LOCAL_SUN4 /usr/local rfs ro 0 0
```

`SunInstall` creates a default `fstab` file for all machines. Its contents depend on the type of machine installed. The first five entries shown above are the NFS mounts a diskless client must perform in order to receive the file systems it needs to operate. These entries are present by default on a diskless client.

The last entry shows an RFS mount, where the client mounts a file hierarchy available from an RFS server. This is an example of an additional file hierarchy that you can have the client mount when it boots. You must become superuser and use your text editor to add new entries to `fstab`. The `fstab` file is more fully described in *The Sun Network File System Service*.

passwd

The `/etc/passwd` file contains the login names, encrypted passwords, and other important identification information for all users (and programs) that are permitted to log in to your machine. Here is a sample `/etc/passwd` file for a diskless client.

```

root:AnLB9ydRMcRhk:0:1:Operator:/:/bin/csh
nobody:*:65534:65534:/:/
daemon:*:1:1:/:/
sys:*:2:2:/:/bin/csh
bin:*:3:3:/:/bin:
uucp:*:4:8:/:var/spool/uucppublic:
news:*:6:6:/:var/spool/news:/bin/csh
ingres:*:7:7:/:usr/ingres:/bin/csh
audit:*:9:9:/:etc/security/audit:/bin/csh
sync:1:1:/:/bin/sync
sysdiag:*:0:1:System Diagnostic:/usr/diag/sysdiag:/usr/diag/sysdiag/sysdiag
stefania:ZRVQ6OQsLUYdM:3747:20:Stephanie:/home/raks/stefania:/bin/csh

```

This is a “local” file, which means its permissions pertain exclusively to your machine. If you are on a network running NIS, you can indicate in the `/etc/passwd` file any (or all) users in the NIS `passwd` maps to whom you want to grant local access to your machine.

The `passwd` database is your system’s first line of security. Its fields are explained fully in in the `passwd(5)` man page. The NIS password map and how it affects access to machines on a network is explained in *The Network Information Service*.

group

The `group` database is a local database listing groups of users on your local machine. Your site might want to organize users into groups by department or by project teams—people who need to access the same information, which should not be accessed by others outside the group.

To edit the `group` database, you log in as superuser and modify the file `/etc/group` with your text editor. `/etc/group` uses the following format:

```

wheel:*:0:
nogroup:*:65534:
daemon:*:1:
kmem:*:2:
bin:*:3:
tty:*:4:
operator:*:5:
news:*:6:
uucp:*:8:
audit:*:9:
staff:*:10:
other:*:20:
+:

```

As is the `passwd` database, the `group` database is a security mechanism. Its fields and its significance to a network client are fully explained in *The Sun Network File System Service* and in the `group(5)` man page. NIS group policies are discussed in *The Network Information Service*.

hosts

The `hosts` database lists the names and IP addresses of machines on the network that the client must know about. Programs on your local machine refer to the `hosts` database to determine its IP address, the address of the NFS file server of which your machine is a client, and other NFS file servers your client needs to know. The local version of the `hosts` database takes the form of the file `/etc/hosts`, as shown below.

```
#
# Host Database
#
# If NIS is running, this file is only
# consulted when booting
#
127.0.0.1      localhost
#
192.9.200.2    raks
192.9.200.100 dancers loghost
```

The numbers in the first column are the IP addresses of the client and the servers that it knows. For example, client `raks` has the IP address `192.9.200.2`. The second column consists of the machines' names. The third column and any subsequent columns contain nicknames for the host. The chapter *The SunOS Network Environment*, and the `hosts(5)` man page contain complete information about the `hosts` database.

hosts.equiv

The `/etc/hosts.equiv` file is a list of host machines whose users are permitted to log in to your machine over the network without supplying a password. It is a local file, pertaining only to your system, which you modify by logging in as superuser and using your text editor to make changes.

A default `/etc/hosts.equiv` file simply contains a plus sign (+). If your machine is on a network running NIS services, this means access is allowed to all others that NIS knows, as described in *The Network Information Service*. Once you edit it, a typical `hosts.equiv` file might have the following structure.

```
host1
host2
+@group1
-@group2
```

The chapter *The SunOS Network Environment* explains how `hosts.equiv` helps to keep your configuration secure in a network environment. Also refer to the `hosts.equiv(5)` man page for more information.

.rhosts

The `.rhosts` file, which is in your home directory, is used for additional permission checking when someone tries to access your machine over the network. It has the same format as `/etc/hosts.equiv`.


```
host1
host2
+@group1
-@group2
```

The SunOS Network Environment contains more information about `.rhosts`.

aliases

The `aliases` database is used by the `sendmail` mail routing program. It has two forms, the `/etc/aliases` file and the `aliases` NIS maps.

You can edit the `/etc/aliases` file when you want to give `sendmail` at least one alternative name, or *alias*, for a user. `sendmail` then recognizes this alias as having the same machine address as the user's official name. Such an entry is shown next.

```
jgoodyear: amina@raks
amina: jgoodyear
```

where the first entry,

```
jgoodyear: amina@raks
```

gives the official user name (`jgoodyear`) and mailing address (`amina@raks`). The second line,

```
amina: jgoodyear
```

tells `sendmail` that the address `jgoodyear` is to be rewritten as `amina@raks`. You can send mail to either `amina` or `jgoodyear`, and it will be delivered to `amina@raks`.

You can also use `/etc/aliases` to create a mail distribution list for a group of people. Consider this list.

```
samba: benny, chris, daria, david, delores, fastfeet, josefina
```

The alias `samba` is the name of a distribution list and the names following are the user names to receive all mail addressed to "samba."

Note that you have to use the `newaliases` command in order for these changes to take affect. The chapter *Administering Electronic Mail* explains `/etc/aliases` and `sendmail` in detail, as does the `aliases(5)` man page.

printcap

The `/etc/printcap` file is a local database describing the types of printers used by your machine. The printer spooling system reads the `printcap` file whenever you send output to a printer, whether it is directly attached to your machine or elsewhere on the network.

Below is a `printcap` file that includes four remote printers.

```

0|pp|printronix|Printronix:
    :lp=:rm=lprhost:rp=printronix:sd=/var/spool/ppd:\
    :lf=/var/spool/ppd/log:
1|vp|versatec|Versatec:
    :lp=:rm=lprhost:rp=versatec:sd=/var/spool/vpd:\
    :lf=/var/spool/vpd/log:
2|ip|imp|240|imagen|Imagen:
    :lp=:rm=lprhost:rp=imagen:sd=/var/spool/ipd:\
    :lf=/var/spool/ipd/log:mx#0:
3|ruby|Ruby's LaserWriter:
    :lp=:rm=ruby:rp=lw:sd=/var/spool/lw/ruby:\
    :lf=/dev/console:mx#0:sf:sb:

```

You will want to modify the `printcap` file to describe the printer attached to your workstation, to add additional printers on your network, and to indicate your default printer. *Adding Hardware to Your System* explains how to administer printers, including how to modify `/etc/printcap`. Also refer to the `printcap(5)` man page for more information.

Files for Administering a Dataless Client

As administrator of a dataless client, you have `/` and `swap` on your local disk but rely on the server for your `/usr` file system. In most respects, your configuration functions on the network similarly to a diskless client. Therefore, you should learn about the databases and local files described above for administering a diskless client, an example of which follows.

```

/var/spool/cron/crontabs/*
/etc/fstab
passwd database
hosts database
/etc/hosts.equiv
.rhosts
group database
aliases database
/etc/printcap

```

If you need to find out more information about the software installed on your disk, you should consult the files in the `/etc/install` directory. This directory is further described in *Files for Administering a Server*. However, because you probably will be responsible for backing up the files on your local disk, you should also understand the `/etc/dumpdates` file described in *Files for Administering a Server*.

Files for Administering a Standalone System

Your standalone configuration has `/`, `/usr`, and `swap` on separate partitions on the local disk. Because standalones have local disk and, possibly, tape drives, they do not need a network to function. If your site does not have a network, you can set up your machine as a single user standalone or as a time-sharing system with terminals attached. Alternatively, you can attach either standalone configuration to a network.

If you administer a non-networked standalone configuration, learn about the following local files and administrative databases, previously described in the

subsection *Files for Administering Diskless Clients*.

```
aliases database
/var/spool/cron/crontabs/*
/etc/fstab
passwd database
group database
/etc/printcap
```

You should also learn about the next files and directories, especially if you have a time-sharing standalone:

```
dumpdates      (for backing up files)
ttytab         (if you plan to attach terminals)
install        (for information about your local configuration)
```

These files are described in the subsection, *Files for Administering a Server*.

If your standalone configuration is on a network, also learn about the following files and databases.

```
hosts database
/etc/hosts.equiv
.rhosts
```

These were described previously in *Files for Administering Diskless Clients*. You should also learn about the `exports` file, which is described in the next subsection.

Files for Administering a Server

As system administrator of a server, you need to learn about *local* files that pertain specifically to your machine, including these.

```
crontab
group
passwd
hosts.equiv
.rhosts
printcap
```

These files are described above in *Files for Administering Diskless Clients*. Other files and directories that affect your system locally follow.

```
dumpdates      (for backing up files)
ttytab         (if you plan to attach a terminal)
install        (for information about your local configuration)
```

They are described in this subsection.

You also need to learn about local files and databases for managing machines and file systems over the network, including these.

```

bootparams database
ethers database
/etc/exports
/etc/fstab
hosts database
netgroup database
networks database

```

They also are introduced in this subsection.

dumpdates

The `/etc/dumpdates` file contains a record of each time you dump (back up) a file system. A typical `/etc/dumpdates` file looks like the following:

```

/dev/rxy0a      0 Tue Aug 18 08:47:53 1987
/dev/rxy0f      0 Sat Jul 18 08:12:49 1987
/dev/rxy0h      0 Sat Jul 18 08:18:34 1987
/dev/rxy0g      0 Sat May 23 08:02:34 1987
/dev/rxy0f      5 Fri Aug 14 08:16:26 1987
/dev/rxy0h      5 Fri Aug 14 08:21:27 1987
/dev/rxy0g      5 Fri May 29 09:03:07 1987
/dev/rxy0f      9 Tue Aug 18 08:48:35 1987
/dev/rxy0h      9 Tue Aug 18 08:52:27 1987
/dev/rxy0g      9 Thu Jun  4 09:21:02 1987
/dev/rxy2c      0 Sat Jul 18 08:38:56 1987

```

`/etc/dumpdates` is fully described in *File Maintenance*.

ttytab

The `/etc/ttytab` file is a database containing descriptions of the terminals attached to your configuration. Here is a segment from an `/etc/ttytab` file.

```

# @(#)ttytab 1.5 88/05/13 SMI
#
# name      getty                type                status  comments
#
console    "/usr/etc/getty Console-9600"  wyse-925           on local secure
ttya       "/usr/etc/getty std.9600"      unknown            off local secure
ttyb       "/usr/etc/getty std.9600"      unknown            off local secure
ttym0     "/usr/etc/getty std.9600"      unknown            off local secure
ttym1     "/usr/etc/getty std.9600"      unknown            off local secure
ttym2     "/usr/etc/getty std.9600"      unknown            off local secure
ttyp0     none                          network            off secure
ttyp1     none                          network            off secure
ttyp2     none                          network            off secure
ttyp3     none                          network            off secure

```

`ttytab` is fully explained in *Adding Hardware to Your System*. Also refer to the `ttytab(5)` man page in the *SunOS Reference Manual*

install

The directory `/etc/install` contains informative files that tell you about your system configuration. `/etc/install` on your system will resemble the following listing.

```
EXCLUDELIST
XDRTOC
appl_media_file.sun3.sunos.4.1
appl_media_file.sun4.sunos.4.1
arch_info
arch_list
category.standalone
client.ariel
client.neo
client.rigel
client.sbuffles
client.tarheel
client_list
client_list.sun3.sun3.sunos.4.1
cmdfile.xy0
cmdfile.xy2
default_client_info
default_sys_info
disk_info.xy0
disk_info.xy0.original
disk_info.xy2
disk_info.xy2.original
disk_list
format.xy0.log
format.xy2.log
media_file.sun3.sun3.sunos.4.1
media_file.sun4.sun4.sunos.4.1
media_file.tmp
media_list
mount_list
soft_info.sun3.sun3.sunos.4.1
soft_info.sun4.sun4.sunos.4.1
suninstall.log
sys_info
```

For example, the file `/etc/install/sys_info` on your system will contain information similar to the following, derived from data supplied to the SunInstall program.

```
hostname=dancers,dancers-ebb
sys_type=server
ether_type=ether_1
ether_name0=ie0,ie1
ip0=192.199.199.198,192.199.199.199
yp_type=slave
domainname=eng.moon.com
op_type=install
reboot=no
rewind=no
arch=arch.str=sun4.sun4.sunos.4.1
root=sd0a
user=sd0g
termttype=sun
timezone=US/Pacific
server=
server_ip=
exec_path=
kvm_path=
```

Administering Workstations contains more information about this directory.

bootparams

The bootparams database is maintained on the client server and represented by the `/etc/bootparams` file and corresponding NIS map. Clients use it during booting. Here is a fragment of a typical `/etc/bootparams` file.

```
samba      root=dancers:/export/root/samba \
           swap=dancers:/export/swap/samba
ballet     root=dancers:/export/root/ballet \
           swap=dancers:/export/swap/ballet
```

The file shows the directories that clients `samba` and `ballet` need to mount from NFS server `dancers` in order to boot: `root` and `swap`. For more information about `/etc/bootparams`, refer to the *Booting Up and Shutting Down Your System* chapter.

exports

The `/etc/exports` file lists the file systems that an NFS file server makes available to host machines on its network. You modify it by using a text editor. When you add a new client machine, `/etc/exports` is automatically updated with the client's `/export/exec` directories and `/home` directory.

Here is a fragment of a typical `/etc/exports` for a Sun-4 NFS file server with Sun-3 clients.

```

/usr
/home
/var/spool/mail
/export/local
/export/exec/sun3.sunos.4.1
/export/exec/kvm/sun3.sunos.4.1
/export/root/aretha      -access=aretha,root=aretha
/export/swap/aretha     -access=aretha,root=aretha

```

The file lists the names of exported file systems and machine names or netgroup names allowed access to them. A *netgroup* is a network-wide group of users allowed access to certain files for security and organizational reasons. */etc/exports* is fully described in *The SunOS Network Environment*. Also refer to the `exports(5)` and `netgroup(5)` man pages for more information.

hosts

The */etc/hosts* file on a file server lists the names and addresses of machines on the network, or network *hosts*, that the server knows.

Below is a fragment of a typical */etc/hosts*.

```

# If the yellow pages is running, this file is only consulted
# when booting
#
127.0.0.1 localhost

192.9.200.1      ballet
192.9.200.2      raks
192.9.200.3      samba

192.9.200.100   dancers loghost
#

```

The numbers in the first column are IP numbers for each machine, which identify its address on a network. The next column contains the machine names of the hosts. The third and subsequent columns contain nicknames for the host.

The chapters *The SunOS Network Environment* and *The Network Information Service* explain */etc/hosts* in detail, as does the `hosts(5)` man page.

fstab

The */etc/fstab* file shows all file systems that your server mounts upon boot up. Every time you boot your system, commands that mount file systems read the *fstab* file. Here is an example of a server's *fstab* file.

```

/dev/xd0a / 4.2 rw,nosuid 1 1
/dev/xd0g / usr 4.2 rw 1 2
/dev/xy0f /export/root 4.2 rw,nosuid 1 3
/dev/xy0e /export/swap 4.2 rw,nosuid 1 4
/dev/xy0h /export/exec 4.2 rw 1 5
/dev/xd1g / home 4.2 rw 1 2

```

Note that the default *fstab* file on a file server lists only the local, 4.2 mounts--directories that are mounted directly from the disk. If there is more than one file

server on your network, you can make your file server a client of the remote server by adding NFS and RFS mounts to your `/etc/fstab` file. The `fstab` file is more fully described in *The Sun Network File System Service*.

ethers

The `ethers` database contains the Ethernet address of hosts on the network. Below is a fragment of the `ethers` database, as represented by the `/etc/ethers` file.

```
8:0:20:1:d5:bb raks
8:0:20:1:29:7  samba
8:0:20:1:5e:e3 ballet
8:0:20:1:1f:73 jazz
8:0:20:1:51:a1 silly
```

The first column contains the Ethernet address. The second column is the machine name. *The SunOS Network Environment* and *The Network Information Service*, and the `ethers(5)` man page explain more about this file and the Ethernet.

networks

The `networks` database lists the names and addresses of the networks that are part of your larger network. Here is a fragment of the `networks` database, as represented by the `/etc/networks` file.

```
#
# Some customer networks
#
loopback          127
some-ether        192.9.200      somether ethernet localnet
#
# Internet networks
#
arpanet           10                arpa
#ucb-ether        46                ucbether
#
# Some local networks
#
backbone          192.9.0
hairnet           192.9.1
tulle-bb          192.9.2          # Tulle backbone (ebb)
fishnet           192.9.3
```

Refer to *The SunOS Network Environment* and the `networks(5)` man page for more information.

netgroup

The `netgroup` database is security-related, similar to the `group` database, except that it is network-wide in scope. `netgroup` contains the names of groups of users who are allowed to remotely log in to a machine, and of machines that are allowed to remote mount a file system on another machine on the network. Here is an excerpt from the `netgroup` database, as represented by the `/etc/netgroup` file.


```
aswan          (raks,amina,dance.com) (masr,shamira,dance.com)
justmachines  (raks,-,dance.com) (ballet,-,dance.com) (samba,-,dance.com)
justpeople    (-,amina,dance.com) (-,stefania,dance.com) (-,ernest,dance.com)
```

Refer to *The Sun Network File System Service* and the `netgroup` man page for more information.

Introducing Networks

This chapter provides a brief introduction to the SunOS network environment. It discusses the following topics.

- Network administration tasks
- Types of networks
- IP addresses
- Entities on the network
- The network administrative domain

A network is a group of machines connected together so that they can transfer information. Any machine attached to a network is called a *host*.

Because computer networks are now pervasive worldwide, it is a good idea to learn the basic concepts in this chapter, even if you administer a diskless client or non-networked standalone. Then you will have a good knowledge base should your site expand to its own network.

Part Three, “Network and Communications Administration,” contains instructions for setting up a network, maintaining servers and clients, and using the SunOS network services. It assumes you are familiar with the concepts in Chapter 3.

3.1. Performing Network Administration Tasks

Networks consist of both hardware connections and software interfaces that allow transfer of information over a communications line. Network hardware includes various types of controllers, cables, and data conversion devices. Network software includes the following.

- High-level services, such as NFS and RFS
- Lower level network protocols, such as TCP and IP, that translate information to be handled by the higher level services
- Network interface drivers, such as the `ie` Ethernet driver, that manage the activities of the communications controllers

Network administration involves managing both software and hardware, not only for a number of computers, but for network media, such as cables and bridge boxes. The following list explains the hardware-related responsibilities you should expect to have, depending upon your configuration. (The chapters in Part

Three explain your software responsibilities.)

Network Client You must attach your own machine, but probably will not perform network assembly and maintenance. Hardware manuals that came with your machine contain instructions for attaching it to the network.

Networked Standalone You must attach your standalone to the network, but probably will not perform network assembly and maintenance. Hardware manuals that came with your machine contain instructions for attaching it to the network. If your standalone functions as a time-sharing system, you also must administer terminals and modems that access your standalone. Therefore, you should familiarize yourself with the hardware manuals for this equipment.

Existing File Server You will probably maintain machines and cables or connections on the network, and add new equipment as needed. If your site's needs expand, you may possibly set up an internetwork. Therefore, read the hardware manuals that came with your equipment, including manuals for peripherals such as printers and modems. Be aware that you will have to handle network hardware problems as they arise.

File Server on New Network

Your site may have individuals who plan and assemble network hardware. In this case, your hardware responsibilities will be maintenance. If you are responsible for assembling the network, refer to the hardware manuals shipped with your server, workstations, and other equipment, for instructions.

3.2. Types of Networks

Networks are classified according to the physical distance that they cover: local area (LAN), medium range or campus area (CAN), and wide area (WAN). This section describes local and wide area networks.

Local Area Networks

Local area networks handle data communications over a physically limited area, such as single buildings. LANs consist of hosts, including network servers, connected together by media such as cables. Because the LAN media itself only can span a short distance, the company purchasing the networking equipment usually owns the local network. When you install Release 4.1, SunInstall assumes that your server and its clients are all attached to the same network.

All Sun computers except standalones must belong to a network to operate, but this network does not have to be a wide or medium area. Therefore, in this manual, the generic term "network" means local area network, unless otherwise specified.

Wide Area Networks

A wide area network includes one or more computers that provide file transfer, message routing, and other services. It also includes a large number of client machines—terminals and other computers—linked together by communications media. These client machines can be close to each other or separated by thousands of miles, hence the designation “wide area.” The transmission medium might be telephone lines or satellite dishes, allowing users on opposite ends of the Earth to send and receive information.

Some organizations own wide area networks that allow other organizations to connect to the network. Common commercial examples of wide-area networks include networks that handle airline ticketing services or that connect automated teller machines from different banking institutions.

By default, SunOs Release 4.1 provides your machine with the ability to connect to two types of wide-area networks: `uucp` and the Internet. In addition, your network can join other wide-area networks. Separately purchased communications software from Sun, such as X.25, enables you to communicate with different wide area networks.

`uucp`

`uucp` is an abbreviation for *UNIX-to-UNIX copy*. It is not really a network, but rather a program that uses the telephone lines to transfer information. `uucp` was originally designed as a facility for copying files from one computer running the UNIX operating system to another also running UNIX.

Today, people not only use the term `uucp` for this file copy capability, but also for a loose connection of computers with modems and known phone numbers. This organization is sometimes called the `uucp` network, but it is not a network in the same sense as other wide-area networks. This is because `uucp` is not based on networking protocols, a concept introduced shortly.

Note also that `uucp` refers only to software, not to the network hardware on which it runs. `uucp` runs on many types of computers and modems, all of which use phone lines as their wide area network media. Sun machines and ASCII terminals at remote sites, such as employees’ homes, can access your site through `uucp`. The *Adding Hardware to Your System* chapter explains how to set up a modem. The *Administering the UUCP System* explains how to set up and maintain `uucp`.

The Internet and TCP/IP

The Internet is a registered wide area network originally developed by DARPA (Defense Advanced Research Projects Agency). Today the Network Information Center (NIC) at SRI International maintains the Internet. Like `uucp`, the Internet is essentially software. It was designed to run on all types of computers and network media. Sun provides Internet capability for all its computers.

The Internet supports communications through a series of network *protocols*. These protocols are formal standards that explain how a network should transfer information. These rules are implemented in programs, interfaces, and daemons. The Internet uses a set of protocols called TCP/IP, explained in detail in the *The SunOS Network Environment*. Networked Sun computers use TCP/IP by default.

3.3. Entities on a Network

A Sun network consists of three major categories of equipment: server(s), clients, and the networking hardware that links them together. It also consists of software that performs various types of services, or simply converts messages from one form of electronic signal to another. The next subsections briefly describe the hardware and software that comprise Sun networks.

Communications Media

The most common communications media for linking Sun machines on a local area network is *Ethernet*. It is a popular local area network technology invented by Xerox Corporation. An Ethernet consists of a system of coaxial cables that your company probably owns. Sun also supports a variant of Ethernet called thin-wire Ethernet, which is available on some desktop machines, such as Sun-3/50s and 3/60s. Alternatively, you can purchase other local area networking technologies from Sun and other sources.

The wide area networks that your site can access through SunOS software use long-haul networking media for linking hosts. `uucp` uses telephone lines. The Internet originally used the ARPANET long-haul network, but now runs on a collection of regional networks and a “backbone” network called NSFnet.

Server

As mentioned in Chapter 1, a machine is considered a server if it provides a network service, such as disk storage, file service, electronic mail, or print service. Additionally, the daemons that handle these network operations on the server machines are sometimes referred to as *server processes*. Be careful not to confuse the two.

Client

In a Sun network environment, client workstations fall into three categories: diskless, dataless, or networked-standalone, as described in the first two chapters. In addition, client machines run daemons that use resources provided by the server processes. These daemons are called *client processes* and shouldn't be confused with client machines.

Modem

A modem is one type of electronic device that you can use to enable remote communications between a Sun workstation or terminal and a computer at a remote distance. To use `uucp`, you connect the Sun workstation to a modem, which you then attach to a telephone line.

Network Services

Sun server machines are categorized by the network services they provide in these four areas.

- File service and disk storage

These services are provided by *NFS file servers* or *RFS file servers*, depending on which service the particular machine provides. In Release 4.1, the NFS file server also provides the files that diskless and dataless clients require to boot and operate. The chapter *The Sun Network File System Service* explains how NFS works and how to set up and maintain this service on your system. The chapter *The Remote File Sharing Service* explains RFS installation, operation, and maintenance.

- Name services

These services provide network-wide information about users, host machines, and other networks. For local area networks running NFS, you can implement name service by setting up NIS servers. If your local area network runs RFS, RFS name servers provide a limited form of name service. To implement name service across two or more linked networks, you use Domain Name Service (DNS), as described in the *Administering Domain Name Service* chapter.

- Electronic Mail forwarding

Mail servers provide this service. Electronic mail service enables users on a network to exchange messages. The *Administering Electronic Mail* chapter explains how to set up and maintain this service.

- Printer Service

This service is provided by a *print server*. The chapters *Adding Hardware to Your System* and *Maintaining Printers and Print Servers* explain how to add printers to a machine and maintain it as a print server.

3.4. The Concept of Administrative Domains

Within the Sun network environment, the overall term *domain* means entities on a network grouped together for administrative purposes. The domain concept appears in three different contexts.

- The *network administrative domain*, which enables you to administer a group of hosts as a single entity. (See Chapter 13 for a full description.)
- The *RFS domain*, which you set up for machines within a network administrative domain that use the Remote File Sharing service. (See Chapter 18 for a full description.)
- The *NIS domain*, which is a series of databases maintained on a network with NIS services. (See Chapter 16 for a full description.)

In all cases, the reason for domains is essentially the same: it simplifies administration of large networks. For example, you can add a user to a large group of machines with one operation, instead of adding the user to each machine separately.

Although the administrative concept of a domain can be independent of the way machines are connected together, you probably will find it convenient to have administrative boundaries correspond to physical or geographical boundaries. If you have a small site, you might have a single administrative domain and a single local area network. Larger sites might have two or more local area networks that communicate with each other through a mechanism known as a router, forming an *internetwork*. If all networks in the internetwork are under a single administration, you can group them together under a single domain. Even larger sites might need both multiple networks and multiple domains.

3.5. Obtaining an IP Address

SunInstall requires you to supply an Internet address, or *IP address*, for your machine during the Release 4.1 installation process. TCP/IP uses the IP address as a method for identifying and locating a host on the network.

Before you run SunInstall and attach hosts to an existing network, you should obtain an IP address for your server or clients from the network administrator. If you are setting up a new network, you must register your network and obtain an IP address for your network from the NIC before running SunInstall. Do this regardless of whether or not your site will join the actual Internet. The chapter *The SunOS Network Environment*, and the appendices *IP Number Registration Form* and *Internet Domain Registration Form* have complete information about contacting the NIC, including forms required to get an IP number and to register your network.

Glossary of Terms

architecture	The specific components of a computer system and the way they interact with one another. From a Sun kernel perspective, “architecture” refers to the type of CPU chip in the computer. A Sun-3 machine is said to have 68020 architecture, because it contains this CPU chip. A Sun-3x has 68030 architecture. Sun-4s and SPARCsystems have SPARC architecture, because they contain a SPARC CPU chip.
ARP	The ARP (Address Resolution Protocol) maps an IP address to its corresponding Ethernet address.
ARPANET	The Advanced Research Projects Agency funded network backbone, for which TCP/IP was originally developed. (See also DoD Internet, below)
BBN	Bolt, Beranek, and Newman, a private company which offers network management services.
binding	The process during which a client finds out where a server is so that the client can receive services. NFS binding is explicitly set up by the user and remains in effect until the user terminates the bind, for example by modifying the <code>/etc/fstab</code> file. NIS binding occurs when a client’s request is answered by a server and is terminated when the server no longer responds.
boot block	An 8 KB disk block that contains information used during booting: block numbers pointing to the location of the <code>/boot</code> program on that disk. The boot block directly follows the disk label.
booting	The process of powering up the computer, testing to determine which attached hardware devices are running, and bringing the operating system kernel into memory and operation.
bridge	A communications device that selectively copies packets between networks of the same type.
bus	A cable or circuit used for the transfer of data or electrical signals among devices.

caching-only server	A domain name server that is not authoritative for any domain. This server queries servers who have authority for the information needed and caches that data.
client (dataless)	A machine on a network that has its own disk and individual root and swap partition, but relies on the NFS server for booting single and multiuser, and for other services.
client (diskless)	A machine on a network that does not have a disk and relies on the NFS server for file storage and other basic services.
configuration, equipment	The combination of CPU, peripherals, and software, and the way they are interconnected to form a system.
controller	An integrated circuit board that controls the operation of another device or system, such as a graphics board that controls a graphics color monitor.
daemon	A continuously running process that handles system-wide functions, such as network administration or line printer spooling.
DARPA Internet	The Defense Advanced Research Projects Agency Internet (see also Internet).
datagram	Transmission unit used by the TCP/IP network protocol suite at the IP level.
DDN Internet	The Defense Data Network Internet (see also Internet).
device	A hardware component acting as a unit that performs a specific function, such as formatting and printing output (a printer) or reading and writing information on a disk (a disk drive). The SunOS operating system treats all devices as files.
device driver	A program within the kernel that controls the operation of devices. For example, the operation of the SCSI disk controller is handled by a disk device driver.
DoD Internet	The Department of Defense Internet, a wide area network to which the NSFnet and ARPANET belong (see also, Internet).
domain	A name given to a group of machines administered together. In NIS terminology, a domain is a group of machines that access the same NIS maps. In RFS terminology, a domain is a group of machines running RFS software that are served by particular set of RFS name servers.
Domain Name Service	The name service of the TCP/IP protocol family, which provides information about machines on networks both locally and at great distances.
dump	The process of copying directories onto a tape for offline storage, using the dump command.

EGP	Exterior Gateway Protocol, a specialized protocol that allows exchange of information with a backbone network under a separate administration.
Ethernet	A commonly-used local area network technology originally developed by Xerox Corporation.
export	The process by which a server advertises the file systems that it allows hosts on a network to access.
file system	A hierarchical arrangement of directories and files.
gateway	A device that enables networks using different protocols to communicate with each other.
global file	A file containing information such as user, host, and network names, that is network wide in scope.
head	The mechanism on a disk drive that reads and writes information on a disk.
header	Information attached to the beginning of data. Headers usually contain information about the following data to aid in processing it.
heterogeneous server	An NFS server that has clients both of its own architecture and other architectures. For example, a Sun-4 server that has Sun-3 and Sun-3x clients is a heterogeneous server.
homogeneous server	An NFS server that has clients only of its own architecture.
host	A computer attached to a network.
ICMP	Internet Control Message Protocol, which is responsible for handling errors and printing error messages.
inode	An entry in a pre-designated area of a disk that describes where a file is located on that disk, the file's size, when it was last used, and other identification information.
interface, hardware network	A hardware connection to a network, such as an Ethernet board.
Internet	A world-wide wide area network using TCP/IP protocol that was originally sponsored by the Defense Advanced Research Project Agency (DARPA). The term Internet is commonly used to refer to any and all of ARPANET, DARPANET, DDN, or DoD Internets.
internetwork	A group of networks interconnected with routers, which uses the TCP/IP protocol.

IP	Internet Protocol, which allows host-to-host datagram delivery.
IP address	A unique number that identifies each host in a network.
IP network number	A unique number which identifies each IP network (See IP address.)
kernel	The master program set of SunOS software that manages all the physical resources of the computer, including file system management, virtual memory, reading and writing files to disks and tapes, scheduling of processes, printing, and communicating over a network.
label	Information written by the <code>format</code> program starting at cylinder 0 of a disk. The disk label describes the size and boundaries of the disk's partitions and its disk type.
library routines	A series of SunOS functions that can be called by user programs written in C and other compatible programming languages.
local file	A file containing information specific to the machine where it resides. When using NIS, the local file is checked first before a corresponding global file is checked.
map	A file used by NIS that holds information of a particular type, for example, the password entries of all users on a network or the names of all host machines on a network.
makefile	A file used by the <code>make</code> command, which describes files that <code>make</code> must process and programs that <code>make</code> must run.
modem	An electronic device that you can use to connect a Sun workstation to a telephone line.
mount	The process of accessing a directory from a disk attached to the machine making the mount request (4.2 mount) or remote disk on a network (NFS or RFS mount).
netgroup	A network-wide group of machines granted identical access to certain network resources for security and organizational reasons.
network (local area)	A network consisting of a number of machines that share resources such as files and mail. The network covers a physically limited area no greater than two miles.
network (wide area)	A network consisting of one or more large computers providing services such as file transfer, and a large number of client computers that use the services. This network may cover a large physical area, sometimes spanning the globe.

Network File System Service (NFS)	A network service that provides file sharing among hosts. NFS servers also provide kernels and swap files to diskless clients so that they can boot up.
network mask	A number used by software to separate the local subnet address from the rest of a given IP address.
network protocols	Sets of rules that explain how software and hardware should interact within a network to transmit information.
network number	A number that the NIC assigns to your network. The net number forms the first part of a host's IP address.
NIC	The NIC (Network Information Center) is the service run by Stanford Research Institute (SRI) that administers IP network numbers and domain names.
NIC handle	A unique NIC database identifier assigned to a network's administrator, technical contact, etc.
NSFnet	The medium-area network media that the Internet software now runs on.
packet	A group of information in a fixed format that is transmitted as a unit over communications lines.
packet switching	A concept wherein a network transmits packets over connections that last only for the duration of the transmission.
page	A standard unit of memory with an architecture-dependent size that is the smallest entity manipulated by the kernel's virtual memory system.
platter	A flat disk made of magnetic media that is mounted on a spindle. A disk such as those used by Sun computers actually is composed of a number of platters.
partition	A discrete portion of a disk, configured during installation.
port numbers	Numbers used by TCP/IP protocols to identify the end points of communication.
primary master server (DNS)	The primary Domain Name server for a zone, which maintains all the data corresponding to its domain.
process	A program in operation. For example, a daemon is a system process that is always running on the system.
protocol	A set of formal rules explaining how hardware and software on a network should interact in order to transmit information. The default protocol for SunOS networks is TCP/IP.

pseudo-device	Software subsystems or drivers with no associated hardware.
RARP	Reverse Address Resolution Protocol is the reverse of ARP, that is, it maps Ethernet addresses to its corresponding IP addresses.
Remote File Sharing Service (RFS)	A network service that provides file sharing and name-to-address mapping in the environment of the RFS administrative domain.
RFS domain	A group of machines running RFS that use the resources of a single master RFS name server.
remote procedure call (RPC)	A routine that enables communication between two remote programs.
repeater	A machine that indiscriminately transmits data from one segment of a network to another (contrast with the definition for <i>bridge</i>).
root user name	SunOS user name that grants special privileges to the person who logs in with that ID. If the user can supply the correct password for the root user name, he or she is given superuser privileges for the particular machine.
router	A device that forwards information between two local area networks that run the same network protocol.
secondary master server (DNS)	A backup server that takes over for the primary master server should it become overloaded or inoperable.
sector	A segment of a disk track, which, on Sun systems, holds 512 bytes of data.
server	A machine that provides a network service, such as disk storage and file transfer, or a program that handles such a service.
SRI	Stanford Research Institute, International, a not-for-profit organization that runs the NIC. (See NIC.)
standalone machine	A workstation with its own disk and tape drives that does not rely on a server in order to boot.
subnet	A networking scheme that divides a single logical network into smaller physical networks to simplify routing.
subnet mask	See
subnet number	The part of an Internet address that refers to a specific subnet.
superblock	A block on the disk that contains information about a file system, such as its name, size in blocks, and so on. Each file system has its own superblock.

superuser	A user with special privileges granted if he or she supplies the correct password when logging in as root or using the <code>su</code> command. For example, only the superuser can change the password file and edit major system administration files in <code>/etc</code> .
symbolic link	A file that consists of a reference to the name of another file. The kernel translates accesses to the symbolic link into accesses to the file it refers to.
TCP	Transmission Control Protocol, a major, reliable data transmitting protocol that is part of the TCP/IP (Internet) protocol suite.
TCP/IP	The protocol suite originally developed for the Internet; it is also called the Internet Protocol suite. SunOS networks run on TCP/IP by default.
time-sharing system	A Sun computer with terminals attached to its serial ports. The terminals rely on the workstation for processing power as well as file service and disk storage.
track	A concentric ring on a disk that passes under a single stationary disk head as the disk rotates.
UDP	User Datagram Protocol, a protocol at the same layer as TCP but without acknowledgment of transmission, and therefore unreliable.
virtual circuit	An apparent connection between processes that is facilitated by TCP. A virtual circuit allows applications to talk to each other as if they had a physical circuit.
virtual memory	A memory management technique used by the operating system for programs that require more space in memory than can be allotted to them. The kernel moves only pages of the program currently needed into memory, while unneeded pages remain on the disk.
NIS	A network service that provides information about machines and services in a local area network.
NIS domain	A master set of NIS maps maintained on the NIS master server and distributed to that server's NIS slaves.
NIS maps	Database-like entities that maintain information about machines on a local area network. Programs that are part of the NIS service query these maps.
zones	Administrative boundaries within a network domain, often made up of one or more sub-domains. (See Domain.)

Part Two: System Administration Procedures

This part contains important procedures and theoretical information for managing all types of Sun configurations. Many procedures are done on a regular basis. However, the frequency with which you perform them, and whether you should perform them at all, depends on your particular configuration.

Who Should Read This Part

Every Sun system administrator should be familiar with the procedures in this part, regardless of their systems' configuration type or level of expertise. Each chapter states the type of configuration to which the procedure applies. In addition, if you are a new system administrator, you may want to refer back to the tables in Chapter 1, which list where to find the basic information you need to learn.

What Is in This Part

NOTE: Always read through an entire section when doing any procedure for the first time. Never try to leap ahead of the given instructions unless you are absolutely certain of your moves, and feel comfortable that you can correct any mistakes you make.

Part Two discusses the following:

- Booting up and shutting down the system.
- Performing file system maintenance, such as checking resource use, backing up and restoring file systems, checking resource use, and setting up the disk quota system.
- Setting up a local security scheme.
- Maintaining individual workstations, such as recovering from crashes, maintaining a system log, and setting up accounting.
- Reconfiguring the kernel, including an annotated description of the GENERIC kernel configuration file for each Sun model.
- Maintaining disks with the `format` program.
- Adding hardware: boards, printers, terminals, modems.
- Administering printers and print servers.

Part Two: System Administration Procedures — *Continued*

Booting Up and Shutting Down Your System

This chapter explains what happens during the booting process and during system shutdown. This discussion is for advanced system administrators who need to know the booting process in detail. The topics include:

- Powering up the machine from the monitor.
- Booting up the system through the automatic boot process.
- Using an alternative boot procedure when automatic boot is interrupted.
- Understanding the `init` daemon and system initialization scripts.
- Stopping the operating system in a safe fashion.

If you are a beginning administrator, try to get an overall understanding of the automatic booting process. Pay close attention to the commands you use to:

- Boot from various devices
- Abort the booting process
- Shut down the system in an orderly fashion
- Boot a client from a remote NFS file server

After you are more familiar with the system, review this chapter to learn booting in greater depth.

For more information on booting, see the *PROM User's Manual* manual regarding the boot PROM and EEPROM, and the man pages regarding programs the booting process uses, particularly `boot(8s)`, `installboot(8s)`, and `init(8)`.

5.1. Changes to `/boot` Prior to 4.x Releases

This section explains changes to the booting process made since Release 4.0. You should read it if you are upgrading a system running a 3.x operating system to Release 4.1. If your Sun computer is brand new with this release, you can skip this section if you wish.

In previous releases booting a standalone workstation or a server from a local disk was done through code in `/boot`, which knew how to access and interpret a file system on that disk. For example, either of the two commands:

```
> b xy(0,0,0)vmunix
```

Or

```
> b sd(0,0,0)vmunix
```

would read the kernel file `/vmunix` from the root partition on a local disk, which might be of type `xy` or `sd`.

Booting a client workstation over the network used code in `/boot`, which knew how to send so-called ND requests over the network, and assumed that there was a server for the client that understood how to turn ND requests into file transfers over the network. For example, the command:

```
> b ie(0,0,0)vmunix
```

would read `/vmunix` from `/pub.MC680x0` on the client's ND server.

In the current release, the following important changes have been made to `/boot`:

- ND code was eliminated since servers no longer support ND operations.
- The program `/boot` now understands how to perform NFS file operations over the network to the client's server.
- Note that the boot program for a client is now located in `/tftpboot/boot.sunx.sunos.osrel` where `x` is 3, 3x, 4, or 4c, depending on the client's architecture.

5.2. Powering Up Self-Test Procedures

The first step in the boot process occurs when you power up your Sun computer. The CPU in the machine has a PROM containing a program generally known as the *monitor*. The monitor controls the operation of the system before the SunOS kernel takes control. The monitor and other aspects of booting, including the programmable EEPROM, are fully discussed in *PROM User's Manual*.

The monitor runs a quick self-test procedure right after you first power on the system. Briefly, the following may happen during selftest:

Critical errors are found. The screen remains dark.

Firmware checks for the existence of a keyboard. However, if the keyboard cannot be found, the monitor checks the values set in the EEPROM, also on the CPU board. If no EEPROM setting specifies where to receive input when no keyboard is found, the monitor then defaults to the serial port.

Connect an ASCII terminal to the serial port. Configure the terminal for 7 bits, even parity, flow control enabled, 9600 baud. Then power on the workstation again and look for messages on the terminal.

No errors are found. The monitor reports this to the screen. The system then begins the automatic boot process.

5.3. The Automatic Boot Process

The automatic boot process is initiated whenever self test completes without error. You can also invoke automatic booting by doing the following:

- Type the `b` command at the monitor prompt (`>`)
- Run the `fastboot` command from the shell (as superuser)
- Run the `reboot` command from the shell (as superuser)

Upon power-up, you should see the following output:

```
Self Test completed successfully
[The customizable banner message]
Auto-boot in progress.
```

When self testing completes or when the monitor receives a `b` command, it immediately attempts to load the standalone boot file `/boot` from a default device. It first tries to locate a Xylogics SMD disk controller, then a SCSI disk controller, and finally tries to load `/boot` over the network.

This sequence of events is controlled by settings in the EEPROM. If you want, you can program the EEPROM so that the monitor automatically tries to boot over any device, or the network, skipping the SMD and SCSI stages.

Booting from a Local Disk

Servers and standalones boot from a local disk by default. When the monitor determines that it should load `/boot` from a disk, it does the following:

1. It loads in a small executable called the *bootblock code* from a known location on the disk. Bootblock code is found at the beginning of each file system on each disk the machine can be booted from. (The root file system partition `sd0a` or `xy0a` contain bootblock code, but other partitions can as well.)
2. The bootblock code reads in `/boot`. The bootblock code knows which blocks in that file system contain the code of `/boot`. This block list is placed in the bootblock code by the `installboot` program, so that the boot block code does not need to interpret the file system structure. As a consequence, any time that you change `/boot` on that file system, you must run `installboot` to reinstall the bootblock code. This is because the list of blocks occupied by `/boot` typically change after each modification. For more information, see the `installboot(8s)` man page.
3. After it is read in, `/boot` first checks to see what device it was read in from. It then attempts to read the kernel file `/vmunix` from the same device.

Note: By convention, the boot program is called `boot`. In reality, you can use any name for the boot program as long as you tell `installboot` about it.

Automatic Booting from Disk

The example below shows the messages displayed on a server or standalone when automatic booting takes place. In the example, responses that you need to type are in **bold face** and comments are in *italics*:

Output from Automatic Booting from Disk

```
> b
Boot: xy(0,0,0)
root on xy0a fstype 4.2
Boot: vmunix
Size: 870120+174744+181872 bytes
```

Here *xy* is the device name of the “best” local disk controller the monitor could find. Other possibilities are *xd* and *sd*.

`/boot` performs a mount operation on the root file system and accesses it with operations similar to the standard kernel VFS/vnode file operations. The result is the same as issuing the command:

```
# mount /dev/sd0a /
```

If no other filename is provided, the file `/vmunix` boots from this disk, by default from Partition A on Drive 0 on Controller 0 for this disk type. The example below illustrates booting of `/vmunix`. Normally the booted file `/vmunix` contains a SunOS kernel. However, it can contain any standalone program that you wish, (for example, a standalone diagnostic program), as long as the disk contains a standard SunOS file system that `/boot` can mount.

Booting over the Network

Diskless clients must boot over the network, using a protocol called Trivial File Transfer Protocol (TFTP). Consequently, remote booting is sometimes referred to as “TFTP booting.”

Booting a New Client over the Network

During SunInstall, the NFS file server automatically copies the boot file `boot.arch` into `/export/exec/kvm/arch/tftpboot`. Thereafter, you have to perform certain operations to boot a new client of an NFS server. These procedures are necessary before normal TFTP booting can occur.

Power up the client and type the following

```
> b ie()
```

For systems that have a Lance Ethernet, replace `ie`, which represents the Ethernet controller, with `le`, to represent the Lance Ethernet controller that those machines use.

The TFTP Boot Process

Here is a summary of the normal TFTP boot process that occurs after you initially boot a client over the network.

1. When the monitor on the client machine determines that it must boot over the network, it broadcasts a request called a REVARP request. This request, which includes the client’s Ethernet address, is done to find out the client’s IP address.

2. On a network without NIS, a RARPD daemon on the server where the client was configured during SunInstall or `add_client` responds by sending the client's IP address. On a network with NIS, any server can consult its maps and send the client its IP address.
3. Upon receiving its IP address, the client broadcasts a TFTP request for its boot program over the network.
4. The server looks in its `/tftpboot` directory for a filename that is the hexadecimal representation of the client's IP address, plus a suffix representing the client's architecture, except for Sun-3s. For example, these filenames might look like:

<i>Filename</i>	<i>Architecture Type</i>
81903225.SUN3	Sun-3
C0095A3A.SUN4	Sun-4

This file is a symbolic link to the client's boot program, typically `boot.sunx.os.rel`, where *x* is either 3, 3x, 4, or 4c, depending on the client's architecture. The `boot.sunx.os.rel` file contains the boot program for a client of architecture *x*.

6. The server then sends the appropriate `/tftpboot/boot.sunx.sunos.osrel` program over the network to the client. Here *x* represents the machine's kernel architecture, sun 3, 3x, 4, or 4c.
7. The client then executes this boot program.
Servers that have no entry for the client in their `/tftpboot` directories will wait a short while before replying, so as not to overwhelm the client with error messages.
8. After the monitor reads it in, `/tftpboot/boot.sunx.sunos.osrel` first checks to see what device it was read in from. It then attempts to read the kernel file `/vmunix` from the same device.

The bootparamd Daemon and /etc/bootparams File

The `bootparamd` daemon and `/etc/bootparams` file assist client machines in finding their IP addresses and the paths of their root and swap directories. After it is loaded, the client's `boot.sunx.os.rel` program broadcasts a REVARP request to find its own IP address, then broadcasts a `whoami` request. A `bootparamd` daemon on a server responds by looking up the client's IP address and returning it to the client's hostname. In the example below, the client's IP address is 192.9.1.91 and its hostname is `solntze`. The client then sends a `get-file` request along with its hostname to its `bootparams` server, which should have an entry for this client in the `bootparams` database.

For example, the server's `/etc/bootparams` file should have the following entry for host `solntze`:

```
solntze      root=estale:/export/root/solntze \  
            swap=estale:/export/swap/solntze
```

This example line from `/etc/bootparams` indicates that client *solntze* should mount its root file system from the directory `/export/root/solntze` on server *estale*.

The `bootparamd` daemon also sends the client the IP address (this is obtained only when NIS is running) of the server where the client should mount its root file system from. This is not necessarily the same server on which the `bootparamd` daemon is running. The root file system should, of course, contain whatever `vmunix` file the client should load. For example, the pathname of the `vmunix` file on the server for client *solntze* is `/export/root/solntze/vmunix`. You can set up `/etc/bootparams` to have `boot.sunx.os.rel` mount any sensible pathname as the root for the client *solntze*.

If you have set up `/etc/bootparams` correctly, then in response to the `get-file` request, `boot.sunx.os.rel`, running on the client, receives the following information:

```
server name: estale  
server IP address: 192.9.1.3  
server root path: /export/root/solntze
```

Now `/boot.sunx.os.rel` can issue a standard mount request to the `rpc.mountd` daemon running on server *estale*. The result is the same as if you had issued the command:

```
# mount estale:/export/root/solntze /
```

Assuming that this mount is successful, `boot.sunx.os.rel` then attempts to open and read the file `/vmunix` in the NFS file system just mounted. Finally, if this is successful, `/boot.sunx.os.rel` downloads `/vmunix` and finally starts its execution. As above, `/vmunix` need not actually contain the kernel; it can be any bootable file.

The example below shows messages displayed on the client machine's screen during booting. Commands shown in bold face represent information that you have to type during the boot process.

Autoboot Output When Booting from NFS

```

> b
Boot: ie(0,0,0)
Using IP Address 192.9.1.91 = C809015B
Booting from tftp server at 192.9.1.3 = C8090103
Downloaded 107016 bytes from tftp server.

Using IP Address 192.9.1.91 = C809015B
hostname: solntze
domainname: sweng.sun.com
server name 'estale'
root pathname '/export/root/solntze'
root on estale:/export/root/solntze fstype nfs
Boot: vmunix
Size: 870120+174744+181872 bytes

```

In this network example, the monitor recognizes that there is no local disk and instead loads `/boot.sunx.os.rel` from a server on the network.

5.4. Aborting a Booting Sequence

Occasionally you may have to abort the booting process. Typically you do this when you want to boot from a file in a location other than the default disk partition or NFS file system. Strictly speaking, the automatic boot process only takes place on power-up, or if you type a `b` command with no more parameters to the PROM monitor. However, if you are in the monitor after issuing a `fasthalt` command, or after a crash, the machine is already in the same state as if you had aborted an automatic boot.

The specific abort key sequence depends on your keyboard type. For a Sun-4 keyboard, the sequence is **[Stop-A]**. For a Sun-3 keyboard, the sequence is **[L1-A]**. For a standard terminal keyboard (console only!) the **[BREAK]** key generates an abort.

When you abort the boot process, the monitor displays the message:

```
Abort at xxxxxxxx
```

where `xxxxxxx` is the address the CPU was currently at when you generated the abort sequence. The monitor then displays its `>` prompt and waits for you to type a command. The complete set of commands supported by your monitor depends on its exact revision level. The booting-related commands listed in this section are supported by all revisions of the monitor PROM.

5.5. Booting from Specific Devices

Your Sun machine can boot from:

- Any logical partition of a local disk
- An NFS file system whose name is supplied by the `bootparamd` daemon
- The first file of a local tape drive

As mentioned previously, the monitor automatically attempts to boot `vmunix` from a default local disk partition or NFS file system. To boot a different file or a

file from a device other than the default, abort the automatic boot process, then issue the appropriate version of the monitor boot command.

If you are a beginning administrator, pay particular attention to the next two subsections on standard booting syntax and on booting single user. The later subsections, which deal with booting from alternate devices, are directed to sophisticated administrators. Under normal circumstances, you do not need to use alternate devices to boot up.

The Standard Booting Syntax

The boot command has the syntax:

```
>b device(parameters)pathname args
```

where *device* is the type of hardware to boot from, *parameters* represents the partition or other information about the device, *pathname* is the name of the actual program to be booted in a file system mounted from that device, and *args* are optional arguments to the program.

To determine which devices your monitor PROM can boot from, type:

```
> b ?
```

to display the list of appropriate devices. The monitor lists them in the same order as they are tried during the automatic boot procedure. Note that the presence of a device in this list does not mean that it is operational or even connected to the current host. The automatic boot process eventually uses the first device on the list that it finds operational.

To boot from the monitor command level on the default device (the first device the monitor PROM can find to boot from), type:

```
> b
```

You normally use this command after interrupting automatic reboot. You also use this command when rebooting after reaching the monitor PROM level by other means. For example, when you issue a `halt` command or if your system has crashed, you should use this command to reboot `/vmunix` on the default file system. The result is the same as at automatic reboot.

Booting Up in Single-User Mode

It is important for you to understand how to boot up a machine to run in single-user mode. In this mode, your machine only mounts the `/` and `/usr` file system. The machine has not yet started any daemons and the `TERM` variable is not set.

You need to boot your machine in single-user mode to fix certain problems, and to run certain programs. When booting multiuser fails, booting single user sometimes works.

To reboot `/vmunix` from the default file system and come up in single-user mode, type:

```
> b -s
```

The system sends various messages as it checks to see if expected devices exist. Then it stops in single-user mode, giving you the pound sign (#) prompt.

While the system is in this mode, you can try to fix it so that it will work in multiuser mode. For instance, if the `/etc/passwd` file is corrupted, no one can log in to the multiuser system. If you boot single user, you can edit the `/etc/passwd` file from your backup copy and reboot multiuser.

As another example, the file system check program `fsck` may fail. `fsck` runs as part of automatic reboot. You can fix the file system by rerunning `fsck` in single user mode.

Finally, there are certain times when you need to run a command while in single-user mode. For example, you should run `/usr/etc/dump` while the system is in single-user mode, whenever you do full system backups.

When you have completed your tasks in single-user mode, bring the system up in multiuser mode by typing the **CTRL** key and the `D` key simultaneously:

```
# ^D
```

If for some reason your machine cannot access its local file system, you may have to boot a program from the network or from a tape. For example, if your local disk becomes corrupted, you can boot `MUNIX` from tape, as described in *Installing SunOS 4.1*. From there you can run the `format` disk formatting program from tape or from an NFS server to fix your disk. (See *Installing SunOS 4.1* and the chapter *Maintaining Disks* for more information.)

5.6. The `init` Daemon and System Initialization Scripts

The `init` daemon runs as the last step in the booting process. It invokes the system initialization scripts `/etc/rc.boot`, `/etc/rc` and `/etc/rc.local`. Here is the sequence of events started by `init`.

1. `init` runs `rc.boot`.
2. `rc.boot` sets the machine's name. Then, if the system is to come up multiuser, and `fastboot` or `fasthalt` were not run, it invokes `fsck` with the `preen, p` option.
3. `fsck` checks the disk for inconsistencies.
4. If `fsck` does not report problems, `init` invokes `rc`. If `fsck` detects a serious problem, `init` brings the system up in single user mode. When you press **Control-D** to leave single user mode, `rc` is invoked.
5. `rc` mounts file systems on the machine's local disks, if any (4.2 mounts). Then it passes control to `rc.local`.
6. `rc.local` starts daemons on the local machine that handle NFS, NIS, and mail requests. It mounts file systems that the machine accesses over the network (NFS mounts). Finally, it returns control to `rc`.

7. `rc` starts up standard daemons, preserves editor files, and initiates other system activities, such as starting the network or system accounting, if applicable to the currently booting machine.
8. When `rc` finishes running, the system comes up multiuser.

You should go into the `/etc` directory and display the `rc.boot`, `rc`, and `rc.local` scripts to get an overall picture of their contents. You do not have to understand what happens in each line of code. Nevertheless, you should be aware that you may have to log in as root and edit these scripts to modify your system or to fix problems. Later chapters in this manual describe how to make these modifications. Refer also to the `init(8)` and `rc(8)` man pages for more information.

5.7. Shutdown Procedures

The operating system provides several commands for shutting down a system in a non-emergency situation. These commands include:

```
/usr/etc/shutdown
/usr/etc/halt
/usr/etc/reboot
/usr/etc/fastboot
```

You must be superuser to run any of them.

The shutdown Command

The `/usr/etc/shutdown` command provides an automated shutdown procedure that notifies users that a system halt is pending. Its syntax is:

```
# /usr/etc/shutdown [ -fhknr ] [ time [warning-message ... ]
```

Use `shutdown` for shutting down a system with multiple users, that is, a server or standalone used as a time-sharing system. You may optionally specify a time to the `time` argument and a message to be broadcast to all current users for the `message` argument. In addition, the various option flags enable you to request that `halt` or `reboot` run automatically after `shutdown` completes. Refer to the man page `shutdown(8)` for a complete description of all options.

`shutdown` inhibits logins and automatically executes `sync` to ensure that all information is written to the disk. At the time you designate, `shutdown` brings the system down to single-user mode.

The halt Command

When you run the `/usr/etc/halt` command, it immediately shuts down the system in an orderly fashion, unless you explicitly specify otherwise. Its syntax is:

```
# /usr/etc/halt [ -nqy ]
```

Refer to the `halt(8)` man page for a complete description of the command's arguments.

`halt` does not have a facility for leaving messages or for specifying a time when it should execute. Therefore, you should use `halt` to shut down diskless and dataless clients, and standalones that are not used for time sharing. If you need to

bring down a server or time-sharing standalone quickly and unexpectedly, you also should use `halt`.

`halt` writes out information to the disks and halts the processor — no warning, no delay. It takes you out of the operating system and back to the monitor. Do not use `halt` on a server instead of `shutdown` unless an immediate system halt is necessary, because users may find this very disturbing.

The `reboot` Command

When the operating system is running and you want to reboot, you normally use `shutdown` on a server or time-shared standalone. However, you might prefer to use `/usr/etc/reboot` on a machine with only one user, or on a server or time-shared standalone currently in single-user mode.

The syntax of `reboot` is:

```
# /usr/etc/reboot [ -dnq ] [ boot_args ]
```

Refer to the `reboot(8)` man page for a complete description of the command's arguments.

When you run `reboot`, it executes `sync` to write out information to disk, then initiates a multiuser reboot. During this process, `fsck` runs and performs its disk checks. If no problems occur, the system comes up multiuser.

The `fastboot` and `fasthalt` Commands

Both `/usr/etc/etc/fastboot` and `/usr/etc/fasthalt` are shell scripts that respectively reboot or halt the system, and arrange that the file systems are not checked when the system comes back up. The syntax diagrams for these commands are:

```
# /usr/etc/fastboot [ boot-options ]
```

Or

```
# /usr/etc/fasthalt [ halt-options ]
```

Refer to the man page `fastboot(8)` for more information.

You should use these commands very carefully because `fsck` will not run when the system reboots.

File Maintenance

This chapter explains the following major tasks that you must perform to administer file systems.

- Backing up and restoring file systems
- Checking disk usage when file systems are full
- Setting up a disk quota system

6.1. Backing Up File Systems

Note:In SunOS terminology, the word *dump* is often used to mean "back up".

This section explains how to back up file systems using the `dump(8)` command. It discusses the following topics:

- Choosing which SunOS file systems to back up
- Planning a backup strategy
- Using tapes and tape controllers for backup
- Using the `dump` command
- Planning an archive strategy
- Planning how to dump various system configurations

File Systems That Should Be Backed Up

Backup is one of the most crucial administration functions that you perform. You must plan and carry out a procedure for regularly scheduled backups of your file systems for two major reasons: to ensure file system integrity against possible system crash, and to ensure user files against accidental deletion. When you back up file systems as scheduled, you have the assurance that you can restore anyone's files to a reasonably recent state.

The SunOS operating system provides two commands for backing up files: `tar` and `dump`. You used `tar` when you loaded the 4.1 operating system. However, because `tar` performs backups on a file-by-file basis, it is not recommended for regular backups. (See the discussion on `tar` at the end of this section.) `dump` backs up files on a per-file-system basis, as you will see in the subsection discussing `dump` syntax. Therefore, when determining your backup strategy, you need to determine which file systems to dump and on what schedule.

Who Needs to Back Up?

You must perform dumps if you administer a machine with a local disk, especially a file server. If you administer a diskless or dataless client, you typically rely on the administrator of the file server to dump your file systems, particularly your home directories.

File Systems to Dump on a Standalone

By default, `suninstall` creates the following file systems on the disk:

Table 6-1 *Default File Systems on a Standalone*

Directory	Partition
/	Partition a
/usr	Partition g
/home	Partition h

The / file system on a standalone system contains `vmunix`, your SunOS kernel, and other important files. It also contains the directory `/var/spool/mail`, where mail files are kept. Therefore, you should back it up at regular intervals, especially if your standalone system functions as a time-sharing system or is networked.

The `/usr` file system contains the important UNIX commands.

NOTE *On systems with disks smaller than 110Mb, that only have a small disk, the default partitions in `suninstall` are / and /usr. Home directories physically reside in `/usr/export/home` and have symbolic links to `/home/machine-name/username`.*

`/home` contains the home directories and subdirectories of all the users on your standalone system. Because user-created files change constantly, you should back up `/home` with much more frequency than you do / and `/usr`, perhaps as often as once a day, depending on your site's requirements.

During `suninstall`, you may have assigned additional file systems such as `/export`, or `/var`, or site specific file systems on other available partitions. Be aware of the partitions where your file systems are located, because you specify the file system to the dump command by its partition name or mount point.

To see what partition a file system is located in, you can check the `/etc/fstab` file, use the `df` command, or the `mount` command (described later in this chapter).

File Systems to Dump on a Server

On a file server, you have to dump not only the file systems that contain the operating system itself, but the file systems for the individual users, as well.

These file systems are, by default:

Table 6-2 *Default File Systems on a Server*

Directory	Partition
/	Partition a
/usr	Partition g
/home	Partition h
/export/swap	Partition e
/export	Partition d

You need to back up periodically the file systems containing the kernel, major commands, and executables (`/`, `/usr`, and `/export`). You should dump these file systems at intervals from once a day to once a month, depending on your site's requirements. For example, if your site frequently adds and removes clients and equipment on the network, you have to change important files in root, including the kernel configuration file. Therefore, you might want to back up root more frequently than if your site seldom changes the network configuration. Furthermore, your site may keep users' mail in the directory `/var/spool/mail` on a mail server, which client machines then mount. If that is the case, you might want to back up root daily to preserve mail. `/usr`'s contents are fairly static and need to be backed up from once a week to once a month.

The root directory of diskless clients are kept in the `/export` file system. Because the information it contains is similar to the server's root directory in partition *a*, it does not change frequently. If your site sends mail to the clients' root directories, you should back up `/export` more frequently.

The file system you need to back up the most frequently is `/home`. Because `/home` contains the home directories and subdirectories of all the users on the system, its files are very volatile. Depending on your site, you should back up `/home` frequently—at the very least once a week, if not daily.

Planning a Dump Strategy

The `dump` command lets you do two kinds of dumps: full dumps and incremental dumps. `dump` provides a system of *levels*, ranging from 0 to 9. With these levels, you can specify whether all files should be dumped or only those that were altered since a lower level dump.

When you specify Level 0 to the `dump` command, it does a *full dump*—a backup of the contents of an entire file system. When you specify Levels 1-9 to `dump`, it does an *incremental dump*—a backup only of files that have changed since the last dump of a lower level. (You will see how to actually specify dump levels in the subsection, *dump Command Syntax*.)

Important Planning Considerations

Plan your dump strategy on paper and consider the following items:

- Which file systems need to be dumped for your particular configuration (standalone or server)?
- How often should you back up? This depends on how many people use your configuration and the type of work that they do. If users constantly create and modify files, consider making daily dumps of `/home` on a server and

standalone systems, especially if they are used for time-sharing and are networked.

- What levels of incremental dumps are necessary so that you can recover files in case a user deletes them or the system crashes? Example dump levels are given in the next subsections.
- How many tapes do you need to implement your backup plan? If cost-effectiveness is a consideration at your site, you might want to devise a strategy that efficiently uses a smaller number of tapes.

Keep in mind that frequent level 0 dumps produce large dump files and use up expensive media. Level 0 dumps also take a long time to write and retrieve, as well as possibly creating many duplicate files when you restore the file system. Therefore, plan incremental dumps to pick up the small changes in the files you need to make finding those files easier.

Here are some hints to help you organize your backup procedure:

- Consider doing incremental dumps every working day, commonly at Level 9. This saves all files modified that day, as well as those files still on disk that were modified since the last dump of a level lower than 9.
- If you continue to do level 9's each day, the dump will grow larger, reflecting the growing number of files changed on the file system since the last lower level dump. If you then do a lower level dump once a week, you save the entire set of changes from the week.

Remember that a file changed on Tuesday, then again on Thursday, goes onto Friday's lower level dump looking like it did Thursday night, not Tuesday night. If a user needs the Tuesday version, you cannot restore it unless you have a Tuesday dump tape (or a Wednesday dump tape, for that matter). Similarly, a file present on Tuesday and Wednesday, but removed on Thursday, will not appear on the Friday lower level dump.

- It is recommended that you save a week's worth of daily, level 9 dumps for at least one week after they were made, if you also have weekly backups at a level lower than the daily level.
- You should probably do a level 0 dump of your root file system once a week to once a month, depending on what your site needs.
- The `dump` command has a `w` and a `W` option that lists the file systems that you have backed up and the file systems that need backing up if you need information to change or check up on your dump strategy. (See the section *dump Command Syntax*).

It is a good idea to periodically clean and check your tape drive for correct operation. A dump tape you cannot use is useless. You can test your tape hardware by copying some files to the tape and reading them back and then comparing the original with the copy. Or you could use the `v` option of the `dump` command. (See the `dump(8)` man page.) Be aware that hardware can fail in ways that the system does not report.

Remember to tailor your strategy for your configuration. Importantly, you must implement a schedule and make sure that tapes that cannot be read due to deterioration or lack of a label are useless. In addition, make sure stored tapes are kept cool and clean and remember to occasionally verify your dump tapes. An unreadable tape may as well not exist.

The next two subsections describe some dump strategies for a file server and a standalone system. Both assume the configuration is heavily used. If your configuration is less-heavily used, you certainly don't have to back it up as often, or use as many tapes.

Backup Strategies for a Server

Below is an example of a backup plan for a file server on a small network where users are doing file-intensive work, such as program development or document production. It assumes a theoretical month that begins on Friday and consists of four, five day work weeks. In practice, you would probably do at least one more Level 9 dump, depending upon the number of days in the month. Also, you might want to schedule end-of-month backups on the last Friday or, if applicable, the last weekend, of the month.

Table 6-3 *Schedule of Backups for a Server*

Directory	Date	Level	Tape Name
/	end-of-month	Level 0	<i>n</i> tapes
/usr	end-of-month	Level 0	"
/export	end-of-month	Level 0	"
/home	end-of-month	Level 0	"
/home	1st Friday	Level 5	Tape A
"	1st Monday	Level 9	Tape B
"	1st Tuesday	Level 9	Tape C
"	1st Wednesday	Level 9	Tape D
"	1st Thursday	Level 9	Tape E
/home	2nd Friday	Level 5	Tape F
"	2nd Monday	Level 9	Tape B
"	2nd Tuesday	Level 9	Tape C
"	2nd Wednesday	Level 9	Tape D
"	2nd Thursday	Level 9	Tape E
/home	3rd Friday	Level 5	Tape G
"	3rd Monday	Level 9	Tape B
"	3rd Tuesday	Level 9	Tape C
"	3rd Wednesday	Level 9	Tape D
"	3rd Thursday	Level 9	Tape E
/home	4th Friday	Level 5	Tape H
"	4th Monday	Level 9	Tape B
"	4th Tuesday	Level 9	Tape C
"	4th Wednesday	Level 9	Tape D
"	4th Thursday	Level 9	Tape E

With this plan, you use *n* tapes (the number of tapes needed for a full backup of

/, /usr, /export, and /home) plus eight additional tapes for the incremental dumps of /home. This plan assumes that each incremental dump uses one tape. Obviously, if you have large file systems that need more than one tape, you will have to calculate how many more tapes you need.

Here is how this plan works:

1. At the end of the month, do a full backup (Level 0) of /, /usr, /export, and /home, and save these tapes for a month. Remember, if you have a small disk, your home directories are probably located under /usr.
2. On the first Friday of the month, do a Level 5 dump of /home, which copies all files changed since the previous lower level dump, in this case the Level 0 dump you did at the end of the month. You use Tape A for this dump, then store it for a month, then use it the first Friday of the next month.
3. On the first Monday of the month, use Tape B to do a Level 9 dump of /home. dump copies all files changed since the previous lower level dump, in this case the Level 5 dump that you did on Friday. Then you store Tape B until the following Monday, when you use it again.
4. On the first Tuesday of the month, use Tape C to do a Level 9 dump of /home. Again, dump copies all files changed since the last lower level dump—Friday's Level 5 dump.
5. Do the Wednesday and Thursday Level 9 dumps onto Tapes D and E.
6. At the end of the week, use Tape F for a Level 5 dump of /home. This tape contains all changes made to files since the Level 0 dump, approximately a week's worth of changes. Save this tape until the second Friday of the next month, when you will use it again.
7. Repeat Steps 3-5 for the next week, and so on until the end of the month, when you once again take full dumps of all file systems.

This plan allows you to save files in their various states for a month. However, it does have certain drawbacks. For example, it assumes that every month starts on Friday and contains only four weeks. The most obvious drawback is that if you use the same tapes every month for the Level 0 backup of home, you would not have a backup copy of a file that a user just deleted but last modified three months earlier. An alternative is to save each Level 0 backup of /home for a year. It requires many tapes, but does ensure that you have a library of tapes to draw upon, in case a user needs to work on an old project that was cancelled and then started up again. Since /, /usr, and /export don't contain files that are modified extensively (unless you use /var/spool/mail to hold mail), you can use the same tapes to safely back them up every month.

A second drawback of the plan above is that the Level 5 dump on the fourth Friday can become quite large. Remember, it's copying all files that were changed since the Level 0 dump a month ago. If the number of tapes you use is a concern at your site, you can consider this alternate plan:

Table 6-4 *An Alternative Backup Schedule for a Server*

Directory	Date	Level
/home	end-of month	Level 0
/home	1st Friday Mon-Thurs	Level 2 Level 9s
/home	2nd Friday Mon-Thurs	Level 3 Level 9s
/home	3rd Friday Mon-Thurs	Level 4 Level 9s
/home	4th Friday Mon-Thurs	Level 5 Level 9s

In this plan, the Level 2 dump on the first Friday copies only those files changed since the last lower level dump—the full backup at the end of the month. The Level 3 dump copies only those files changed since the Level 2 dump the previous week, and so on until the end of the month. Therefore, the Friday dump tapes save only those changes made during a given week, rather than an increasing number of files since the last monthly full dump.

Backup Strategies for a Standalone

As a system administrator of a standalone system, you can use a strategy similar to that pictured in Table 6-3 for the server. The main difference between a server and a standalone system is that on the standalone system, you do not have `/export`. Remember that `/usr` contains the SunOS commands and `/home` the user's home directories. Therefore, a Level 0 dump of `/home` will be large. However, the incremental dumps will only include changes made to home directories, since you probably won't change SunOS files in `/usr` that frequently. If you use the standalone system for time-sharing, `/home` will probably be large from users on terminals.

If your standalone system has a lot of mail activity, you might also want to back up `/` more frequently, perhaps once a week. With this plan, you use n tapes; the number of tapes needed for a full backup of `/`, `/usr` and `/home`. plus eight additional tapes for the incremental dumps of `/home`.

Here is how this plan works:

1. At the end of the month, you do a full backup (Level 0) of `/`, `/usr`, and `/home`. Save these tapes for a month.
2. On the first Friday of the month, you do a Level 5 dump of `/home`, which copies all files changed since the previous lower level dump, in this case the Level 0 dump you did at the end of the month. Use Tape A for this dump, store it for a month, then use it the first Friday of the next month.
3. On the first Monday of the month, you use Tape B to do a Level 9 dump of `/home`. dump copies all files changed since the previous lower level dump, in this case the Level 5 dump that you did on Friday. Then you store Tape B until the following Monday, when you use it again.

4. On the first Tuesday of the month, you use Tape C to do a Level 9 dump of /home. Again, dump copies all files changed since the last lower level dump—Friday's Level 5 dump.
5. Do the Wednesday and Thursday Level 9 dumps onto Tapes D and E.
6. At the end of the week, use Tape F for a Level 5 dump of /home. This tape contains all changes made to files since the Level 0 dump, approximately a week's worth of changes. Save this tape until the second Friday of the next month, when you will use it again.
7. Repeat Steps 3-5 for the next week, and so on until the end of the month, when you once again take full dumps of all file systems.

Tapes and Equipment Used for Backup

You typically back up SunOS configurations using either 1/2 inch reel tape or 1/4 inch cartridge tape. Tape size depends on your tape device and the tape controller board that it is connected to.

Always label your tapes after backup. If you have planned a backup strategy similar to those suggested in the previous subsection, you should indicate on the label Tape A, Tape B, etc. This label should never change. Every time you do a dump, make another tape label containing the backup date, file system backed up, dump level, plus any site-specific information. Store your tapes in a safe location, where they will be free of dust and away from magnetic equipment. Some sites store archived tapes in fire-proof cabinets at remote locations.

Your configuration will have one of the three tape controllers listed below along with its device abbreviation as it appears in the /dev directory, and the width of the tape it supports:

Table 6-5 Sun Tape Controllers

<i>Tape Controller</i>	<i>Device Abbrev</i>	<i>Width</i>
Xylogics 472	mt	1/2 inch
Tapemaster	mt	1/2 inch
SCSI	st	1/4 inch

Sun workstations with tape cartridge units (servers or standalone systems) usually have one of several models of SCSI controllers. Servers with reel-to-reel tape drives usually have a Xylogics 472 controller. Older systems sometimes have Tapemaster controllers.

You have to specify the tape drive name and tape format to the dump command. The following subsections list the specifications for each tape device, grouped by controller. You need to use the device abbreviations shown in these tables when specifying arguments to dump.

You can use the status feature of the mt (1) to figure out what type of tape drive you are using. For example:

1. Load a tape in the drive you want information on.

2. Enter

```
% mt -f /dev/tape status
```

where *tape* is either `rmt0` or `rst0`. A second or third drive, when supported, would be `rst1` and `rst2` respectively.

The status feature reports:

1/2" tape drives:

TapeMaster
Xylogics 472

1/4" tape drives:

Sysgen [QIC-11 and QIC-24]
Emulex MT-02 [QIC-11 and QIC-24]
Archive QIC-150
Wangtek QIC-150

The following example shows what the `mt` command with the status feature reported.

```
% mt -f /dev/rst0 status
Emulex MT-02 QIC-24 tape drive:
sense key(0x2)= not ready residual= 0 retries= 0
file no= 0 block no= 0
```

Specifications for 1/2 Inch Tapes

Devices with 1/2 inch tapes should always have the abbreviation `rmtn` or `nrmtn`, whether you use a Xylogics or Tapemaster controller. Here is what the above designations mean:

- `n` This abbreviation tells `dump` to use the no-rewind device, as discussed below.
- `r` This abbreviation tells `dump` to use the raw device. The term *raw* means, the device does not buffer.
- `mt` This abbreviation tells `dump` that you are using reel-to-reel tape.

When planning your backup, you may decide to store more than one file system on a reel of tape. If so, you need to consider how much data your tape holds, then compare that to the size of the file systems you want to dump. When you copy more than one file system onto a single tape, you **MUST** use `dump`'s no rewind option, (that is, do not rewind the tape after you are done with it), as described in the next subsection. The tables below show tape format in terms of bpi, which means bits-per-inch of data that you can store on the tape. 1600 bits per inch is considered a *low density* format. A 2400 foot tape at 1600 bpi holds approximately 35-40 Mbytes of information. 6250 bits-per-inch is considered a *high density* format. A 2400 foot tape at 6250 bits-per-inch holds approximately 140-150 megabytes of information.

Should you want to store more than one file system on the tape, first use the `df` command to find out the size of the file system (`df` is explained later in this chapter.) Then use the no rewind device with `dump`, (`nrmtn`), as is explained in the section, *Procedures for Dumping Your Configurations*.

Refer to the table below that applies to your tape controller.

Table 6-6 *Specifications for a Xylogics Tape Controller with Fujitsu Tape Unit*

<i>Device Abbrev.</i>	<i>Format</i>	<i>Capacity</i>
<code>rmt0</code>	1600 bpi (rewind)	35-40 Mbytes
<code>nrmt0</code>	1600 bpi (no rewind)	35-40 Mbytes
<code>rmt8</code>	6250 bpi (rewind)	140-150 Mbytes
<code>nrmt8</code>	6250 bpi (no rewind)	140-150 Mbytes

Table 6-7 *Specifications for Tapemaster Controllers and Xylogics/CDC Combination*

<i>Device Abbrev</i>	<i>Tape Length</i>	<i>Capacity</i>
<code>rmt0</code>	1600 bpi (rewind)	35-40 Mbytes
<code>nrmt0</code>	1600 bpi (no rewind)	35-40 Mbytes

Specifications for 1/4 Inch Tapes

Sun computers use various models of SCSI tape controllers for 1/4 inch tape drives. You format tapes in QIC format standard; on Sun equipment, tape drives are either in QIC-11, QIC-24, or QIC-150 format. Usually QIC-11 capable tape drives are 4 track, QIC-24 capable drives are 9 track, and QIC-150 are 18 track, but not always. The tape format, equipment used, tape length, and number of tracks determine how many megabytes of information you can store on the tape. The table below depicts this information. Device abbreviations for these drives are either `rstn` or `nrstn`. Here is what the above designations mean.

- `n` This letter represents an actual number, such as 0, 8, that represents the QIC format of the tape. The table below finds the value of `n` to use for each of these formats.
- `r` This character tells `dump` to use the raw device. The term *raw* means the device does not buffer. You should always specify the raw device when using `dump`.
- `st` This abbreviation indicates tape device `st` for a tape device on the SCSI controller.

When planning your backup, you may decide to copy more than one file system onto a tape cartridge. If so, you need to consider how much data your tape can hold, then compare that to the size of the file systems you want to dump. You need to know tape capacity when using `dump`'s no rewind option, as described in the next subsection.

You format tapes in the QIC format standard; on Sun equipment, tape drives are either in QIC-11, QIC-24, or QIC-150 format. All Sun-3 and Sun-4 1/4 inch tape

drives can read QIC-11 and QIC-24 in 9-track format. Cartridges written on 4-track drives can also be read on 9-track drives, but tapes written on 9-track drives cannot be read past the first 4 tracks on 4-track drives. The rated tape density is the same for both formats.

QIC-150 deviates from this. QIC-150 is another cartridge tape spec of 18-tracks along with a slightly higher tape density. This tape standard maintains compatibility with the older QIC-11 and QIC-24 4- and 9-track tapes in a read-only capability, i.e., QIC-150 drives cannot write QIC-11 nor QIC-24 formats. Also, QIC-150 tapes cannot be read on QIC-11 or QIC-24 drives.

If you need to, you can use the `mt(1)` command to find the tape controller type.

Table 6-8 *Devices and Tapes on a SCSI Tape Controller*

<i>Device Abbrev</i>	<i>Description</i>	<i>Format</i>	<i>Tracks</i>	<i>Tape Length</i>	<i>Capacity</i>
rst0	Sun-3, Sun-4	QIC-11	9	450 ft	45 Mbytes
rst0	Sun-3, Sun-4	QIC-11	9	600 ft	60 Mbytes
rst8	Sun-3, Sun-4	QIC-24	9	450 ft	45 Mbytes
rst8	Sun-3, Sun-4	QIC-24	9	600 ft	60 Mbytes
rst0	Sun-3, Sun-4	QIC-150	18	600 ft	150 Mbytes

When doing a level 0 dump on the tape, first use the `df` command to find out the size of the file system. (`df` is explained later in this chapter.) Then use the `no` rewind device with `dump`, as is explained in the section, *Procedures for Dumping Your Configurations*.

Also see the end of the *Procedures for Dumping to Quarter-Inch Tapes* section for a specific example of using `dump` with an QIC-150 tape.

Using the `dump` Command

You use `dump` in the same fashion whether you are backing up reel-to-reel or cartridge tapes. However, you have to supply your configuration's tape and disk device names as arguments and indicate different options for the types of tape. If you do not remember the proper device abbreviations or other tape-specific information, refer to the tables in the previous section.

dump Command Syntax

The syntax of `dump` is:

```
# /usr/etc/dump options tape_device_name filesystem_to_dump
```

The italicized arguments on the command line are described as follows.

options refers to a series of options that you can supply to `dump`. The options most commonly used are summarized below. Refer to the `dump(8)` man page for a complete description of these options.

- 0-9** Dump level you want to use.
- a** Create a dump table-of-contents archive in a specified file, *archive-file*. `restore` then uses this file to determine

- whether a file is present on a dump tape, and if it is, on which volume it resides. (See `restore` below.)
- c** Dump to cartridge tape. (The default is reel-to-reel.)
 - d** Tape density. The default is 1600 bpi for a 1/2 inch tape and 1000 bpi for a 1/4 inch tape.
 - f** Dump file system to the device indicated later on the command line. The default is `/dev/rmt8`.
 - s** Tells `dump` the size of the tape.
 - u** Write the date the dump was successfully completed to the file `/etc/dumpdates`. (`/etc/dumpdates` is described in the section *Restoring Files and File Systems*.)
 - v** Rewinds the media and checks to verify that the media and the file system are the same. Can use only on a quiescent file system.
 - w** Lists the file systems that need backing up. This information is taken from `/etc/dumpdates` and `/etc/fstab`. When you use this option all other options are ignored and after reporting, `dump` immediately exits.
 - W** Shows all the file systems that appear in `/etc/dumpdates` and highlights those file systems that need backing up.

If you specify the **a** option, then it is a good idea to put it in a secure directory and name the *archive-file* something meaningful like the hostname, date, and dump level; for example, `patches110489-1`. You now have an online list of all the files in the archive that `restore` can use to ensure that you are retrieving the files that you need.

If you specify the **f** option, then you need to specify a value for the argument *tape_device_name*. This represents the device abbreviation for the tape device on your system. If you do not remember the abbreviation, refer to the tables in the previous section, *Tapes and Equipment Used for Backup*. For example, if you have a Xylogics tape controller and dual-density tape drive, you use the following argument to `dump`:

```
/dev/rmt8
```

When you specify the tape device name, you can also type the letter **n** directly before the name, as in

```
/dev/nrst8
```

to indicate “no rewind”. The advantage of specifying no rewind is that you can copy more than one file system onto the tape during a backup procedure. The disadvantage of specifying no rewind is that if you run out of space during the dump, the tape does not rewind before `dump` asks for a new tape. If you have a 1/2 inch tape, you would have to rewind by hand. From the arguments you supply, `dump` automatically calculates how much data it can put onto the tape.

Suppose you load a new tape, then issue a `dump` command. This starts the copy of the file system at the beginning of the tape. If the file system is too large to fit on one tape, `dump` will wait for you to load another tape, then continue the copying process. If you start a `dump` in the middle of a tape, `dump` has no way to know how much tape was already used and cannot compute how much data fits on the rest of the tape. Therefore, before dumping multiple file systems onto one tape of any type, you must calculate the sum, in megabytes, of all the file systems to be dumped. Then organize the order of commands so that the file systems you are dumping will fit on the same tape. Use the `df` command to get the size, in kilobytes, of the file system(s) you plan to dump (when you are doing a level 0 dump).

The argument *filesystem_to_dump* represents the type of disk controller you have and the partition where the file system to be dumped is located. The table below shows the possible disk controllers on your system:

Table 6-9 *Sun Disk Controllers*

<i>Disk controller</i>	<i>Abbreviation</i>
Xylogics 7053	xd
Xylogics 450 & 451	xy
SCSI Disk	sd

The actual format of the *filesystem_to_dump* argument includes the disk controller abbreviation, a number indicating the disk number and the partition letter. The disk number is 0 by default, but may be 1, and so on, depending on the number of disks in your configuration. The partition letter may have the value *a-h*. For example, if you have a Xylogics 7053 on your server and you want to dump `/home` located in partition *h*, specify the argument:

```
/dev/rxd0h
```

If you have a SCSI disk controller on your standalone system (or server) and you want to dump `/usr` located in partition *g*, specify the argument

```
/dev/rsd0g
```

Using Dump to Archive Files

You usually use `dump` for periodic backups, where you save all the files recently modified on a specified filesystem. In addition to normal backups, it is often necessary to archive files that will not be used for some time or to save a snapshot of a set of files.

Examples of situations where you may wish to archive a set of files:

- Archiving the home directory of a former user.
- Saving a snapshot of the source files for a particular release of a software project.

You can use `dump` to perform this archiving function. If you specify a list of files and directories on the `dump` command line instead of an entire filesystem, `dump` saves only those files. Since you can use several other SunOS utilities

such as `tar`, `cpio`, or `bar` to archive files, some guidelines are given here to help you determine your archiving strategy.

Planning Your Archiving Strategy

The features of `dump` that make it useful for file archiving are:

- Ability to handle multiple tape volumes and ability to recover from tape errors which is necessary when archiving very large directories.
- Faster than other SunOS utilities for archiving, especially when writing to tape.
- No limits to filename length. (The utilities `bar` and `tar` limit filenames to 100 characters, and `cpio` limits filenames to 128 characters.)
- Convenient remote tape support.
- Convenient to restore archived files interactively with the `restore` utility.

The limitations to `dump` that may make another SunOS utility more suitable:

- Cannot restore files on the Sun386i or non-Sun systems.
- All the files to be saved must reside on the same filesystem.
- Must have read access to the raw disk partition where the files reside.
- Lacks the file compression feature of `bar`.

Procedures for Dumping Your Configurations

Before you perform the dump procedure, there are several steps you need to take to prepare for the dump. These steps are, for the most part, appropriate for both a server and a standalone system.

1. If you are doing a Level 0 dump, take your system down to single user level. You can do this using either of two methods.

NOTE *It is strongly recommended that you perform dumps in single-user mode or with the file system unmounted. If the file system is active while it is being dumped, it is possible that some data may not be included in the dump. Also, if directory level operations (creating, removing, and renaming files and directories) are in progress while the file system is being dumped, it may not be possible to correctly restore the data from the dump tape, including data that was not "active" at the time of the dump.*

- a. Log in as superuser, then run `shutdown`, as described in *Booting Up and Shutting Down Your System*. Then, reboot and come up single user, as shown below:

```
# /usr/etc/shutdown
```

Various prompts from shutdown appear

```
# /usr/etc/halt
```

```
> b -s
```

If you have a server, this method is preferable, because `shutdown` gives you the option to send messages regarding the imminent shutdown of services to others on the network. (This method also works for standalones in a time-sharing environment.)

- b. Log in as superuser, then kill the `init` daemon, as follows:

```
# kill 1
```

This action automatically brings the system down to single-user mode. You might prefer to do this if your system is a *standalone* system.

2. If you are doing a Level 0 dump on any file system, you should, at single-user mode, enter

```
# /usr/etc/fsck
```

to run the file system checking program. `fsck` checks the file system for consistency in the file system structure, then makes minor repairs. This action ensures that your full dump will be reasonably clean. See the chapter *File System Check Program*.

The following subsections are divided into:

- Procedures for dumping systems using a local 1/2 inch tape unit.
- Procedures for dumping systems using a local 1/4 inch tape.
- Procedures for dumping file systems onto a remote tape drive.

You should refer to the subsection that pertains to the tape drive you use on your server or standalone system.

Procedures for Dumping to Half Inch Tapes

The next instructions are examples for dumping files to a local 1/2 inch tape unit. The examples assume that you are backing up a file server with a 1/2 inch tape unit (`mt 8`) that uses high density, 6250 bits-per-inch tape, and has a disk attached to a Xylogics (`xy`) controller unit. (If your file server has a different disk controller or tape unit, simply substitute the appropriate device abbreviations for those shown in the examples.) The examples also assume that your tape has a length of 2400 feet and that `/`, `/usr`, `/home`, and `/export/root` are on the default disk partitions.

A Sample Full Dump

The next instructions show how to back up the four file systems for which you usually take full dumps on a regular basis, such as once a month. If you are unsure about the syntax used, refer back to the subsection, *Using the dump Command*.

1. While the system is in single-user mode, load the tape into the tape device.
2. Dump the root file system by typing:

```
# /usr/etc/dump 0ufd /dev/nrmt8 6250 /dev/rxy0a
```

Here is a breakdown of the arguments given to `/usr/etc/dump`:

- 0ufd** The **0** specifies a full, Level 0 dump; the **u** tells dump to update the `/etc/dumpdates` file; the **f** tells dump to write to a device to be named as the next argument on the command line; the **d** tells dump to compute tape use based on the tape density supplied.
- /dev/nrmt8** Gives dump information about the device it is dumping to. `/dev/nrmt8` tells dump that it should write to a special (device) file with the following name, in this case, `nrmt8`. The actual name `nrmt8` gives instructions about the tape unit. `n` indicates "use the no rewind option". `r` says treat this unit as a raw device. `mt8` indicates that the device is a reel-to-reel, magnetic tape unit using high density, 6250 bpi tape.
- 6250** The argument for the **d** option that dump uses to compute tape use.
- /dev/rxy0a** Gives dump information about the file system you want to back up. `/dev/rxy0a` is the file name for referencing this partition. (Special files can be anywhere, but historically, they are expected to be in `/dev`.) The actual name `rxy0a` gives information about the device where the file system is located. `r` says treat this unit as a raw device. `xy` indicates that the device is attached to a Xylogics 450 or 451 disk controller. `0` indicates that the file system is on the first disk (disk 0) on the first controller, and `a` indicates that the file system is in partition `a`, the historical location for a server's / file system.

NOTE *In Release 4.1, 0a does not always contain the root file system. For example, the Sun-3/80 and the SPARCstation1 now use `sd6a` and `sd0a` respectively for root if they have an internal disk.*

3. Type the following to dump the file server's `/usr` file system.

```
# /usr/etc/dump 0ufd /dev/nrmt8 6250 /dev/rxy0g
```

The arguments to dump are the same as you use for the root file system, except for the final argument, `rxy0g`. The **g** tells dump to back up partition `g`, which holds `/usr`.

4. Back up the clients' root directories by typing the following:

```
# /usr/etc/dump 0ufd /dev/nrmt8 6250 /dev/rxy0d
```

The only difference between this and the previous commands is in *rxv0d*, which tells *dump* to back up partition *d*, which usually holds the */export/root* file system.

5. Back up the clients' home directories by typing the following:

```
# /usr/etc/dump 0ufd /dev/nrmt8 6250 /dev/rxy0h
```

Here you tell *dump* to copy partition *h*, which holds */home*.

6. Type the following:

```
# /usr/bin/mt offline
```

This command tells the system to rewind the tape unit and take it offline. When the tape is rewound, you can remove it from the tape device. Don't forget to label the tape with the date, file system dumped, and dump level.

A Sample Incremental Dump

The next instructions provide sample procedures for an incremental dump, such as you might take daily. (Unless your site has other requirements, you only need to *dump /home* on a daily basis.) The examples show a Level 9 dump; to take a lower level dump, simply substitute 1-8 for the 9 in the instructions below.

1. While the system is in single-user mode (and you, of course, are logged in as superuser), load the tape into the tape device.
2. Dump the */home* file system by typing:

```
# /usr/etc/dump 9uf /dev/nrmt8 /dev/rxy0h
```

Specifying **9**, tells *dump* copies only those files in the */home* file system that have changed since the last lower level dump, as recorded in the file */etc/dumpdates*.

3. Rewind the tape by typing:

```
# /usr/bin/mt offline
```

Then you can remove the tapes from the tape drive. Don't forget to label the tape.

Procedures for Dumping to Quarter-Inch Tapes

The next instructions are examples for dumping files to a local 1/4 inch tape unit.

The next examples assume that you are backing up a configuration with a 1/4 inch SCSI tape unit (*st8*) that uses nine-track tape and has a disk attached to a SCSI (*sd*) controller unit. The instructions are intended for a standalone system. They also apply to a server with a cartridge tape unit, but you also need to

perform the additional steps for servers, as noted in the text.

A Sample Full Dump

The next instructions show how you would back up the file systems for which you usually take full dumps on a regular basis, such as once a month. (If your configuration has a different disk controller or tape unit, simply substitute the appropriate device abbreviations for those shown in the examples.) If you are unsure about the syntax used, refer back to the subsection, *Using the dump Command*.

Before you perform the dump procedure, there are several steps you need to take to prepare for the dump. These steps are, for the most part, appropriate for both a server and a standalone system.

When planning a full dump, think about how the rewind and no rewind options are used. If you are dumping more than one file system, and you know that they will all fit on the tape, it is safe to use the no rewind option. Your dump will place one file system after the next on the tape. If you are unsure, use the rewind option, but be careful that you do not overwrite files you are dumping.

1. If you are doing a Level 0 dump, take your system down to single user level. You can do this using either of two methods.

NOTE *It is strongly recommended that you perform dumps in single-user mode or with the file system unmounted. If the file system is active while it is being dumped, it is possible that some data may not be included in the dump. Also, if directory level operations (creating, removing, and renaming files and directories) are in progress while the file system is being dumped, it may not be possible to correctly restore the data from the dump tape, including data that was not "active" at the time of the dump.*

- a. Log in as superuser, then run `shutdown`, as described in *Booting Up and Shutting Down Your System*. Then, reboot and come up single user, as shown below:

```
# /usr/etc/shutdown
```

Various prompts from shutdown appear

```
# /usr/etc/halt
```

```
>b -s
```

If you have a server, this method is preferable, because `shutdown` gives you the option to send messages regarding the imminent shutdown of services to others on the network. (This method also works for standalones in a time-sharing environment.)

- b. Log in as superuser, then kill the `init` daemon, as follows:

```
# kill 1
```


This action automatically brings the system down to single-user mode. You might prefer to do this if your system is a *standalone* system.

2. If you are doing a Level 0 dump on any file system, you should, at single-user mode, enter

```
# /usr/etc/fsck
```

to run the file system checking program. `fsck` checks the file system for consistency in the file system structure, then makes minor repairs. This action ensures that your full dump will be reasonably clean. See the chapter *File System Check Program*.

1. While the system is in single user mode, load the tape into the tape device.
2. Dump the `/home` file system by typing:

```
# /usr/etc/dump 0ucfd /dev/rst8 6250 /dev/rsd0h
```

Here is a breakdown of the arguments given to `/usr/etc/dump`:

0ucf	The 0 specifies a full, Level 0 dump; the u tells <code>dump</code> to update the <code>/etc/dumpdates</code> file. The c indicates that you are using cartridge tape. The f tells <code>dump</code> to write to the device to be named later on the command line.
/dev/rst8	Gives <code>dump</code> information about the device it is dumping to. /dev tells <code>dump</code> that it should write to a special (device) file with the following name, in this case, <code>rst8</code> . Note that <code>/home</code> on a standalone system may exceed one tape cartridge. r says treat this unit as a raw device. st8 indicates that the device is a cartridge tape unit attached to a SCSI tape controller.
/dev/rsd0h	Gives <code>dump</code> information about the file system you want to copy. /dev/rsd0h tells <code>dump</code> that it should read from a special (device) file with the following name, in this case, <code>rsd0h</code> . r says treat this unit as a raw device. sd indicates that the device is attached to a SCSI disk controller. 0h indicates that the file system is on the first disk (disk 0) on the first controller, and is in partition <i>h</i> , the usual location for <code>/home</code> .

3. Type the following to dump the standalone system's `/` file system.

```
# /usr/etc/dump 0ucf /dev/nrst8 /dev/rsd0a
```

The arguments to `dump` are the same as you use for the `/usr` file system, except for the final argument, `rsd0a`. The **a** tells `dump` to back up partition *a*, which holds `/`.

4. *If your system is a standalone system,*
type the following:

```
# /usr/bin/mt -f /dev/rst8 offline
```

This command tells the system to rewind the tape unit. When the tape is rewound, you can remove it from the tape device. Don't forget to label the tape with the backup date, file system(s) dumped, and the dump level.

If your system is a file server,
continue with the following steps.

5. Back up the clients' root directories by typing the following:

```
# /usr/etc/dump 0ucf /dev/nrst8 /dev/rsd0d
```

The only difference between this and the previous commands is in *rsd0d*, which tells dump to back up partition *d*, which usually holds the */export/root* file system.

6. Back up the clients' home directories by typing the following:

```
# /usr/etc/dump 0ucf /dev/nrst8 /dev/rsd0h
```

Here you've told dump to copy partition *h*, which holds */home*.

7. Go back to Step 4 of this procedure, and follow its instructions for rewinding tape.

A Sample Incremental Dump

The next instructions provide sample procedures for an incremental dump, such as you might take daily. It assumes that you are dumping a standalone system with the users' home directories in the */usr* file system. If your system is a server with a tape cartridge unit, you instead incrementally dump the */home* file system, usually located in partition *h*. The examples show a Level 9 dump; to take a lower level dump, simply substitute 1-8 for the 9 in the instructions below.

1. While the system is in single user mode (and you, of course, are logged in as superuser), load the tape into the tape device.
2. Dump the */usr* file system by typing:

```
# /usr/etc/dump 9ucf /dev/nrst8 /dev/rsd0g
```

Since you specified the arguments **9ucf**, dump copies only those files in the */usr* file system that have changed since the last lower level dump, as recorded in the file */etc/dumpdates*.

For a file server,

use the same command, but change the **g** in `rsd0g` to an **h** as in `rsd0h`, to dump the `/home` file system.

3. Rewind the tape by typing:

```
# /usr/bin/mt -f /dev/rst8 offline
```

Then you can remove the tape from the tape drive. Don't forget to label the tape.

A Sample Dump for Using QIC-150

The following example shows how to use `dump` with QIC-150 tapes.

```
# /usr/etc/dump 0cdtsf 1250 18 570 /dev/rst0 /dev/r_disk_partition_name
```

The arguments on the command line are described as follows.

0	The dump level; whatever level you want to use. Optionally, the u may be used to specify updating of the 'dumpdates' file.
c	Cartridge tape - allows specification of the t (tracks) and s (size) parameters in the 1/4" tape context. Also sets the blocking factor to be 126.
d	Density - 1250 bytes per inch.
t	Tracks - 18.
s	Size - 600 feet less the usual slop factor.
f	Output file name - <code>/dev/rst0</code> . The QIC-150 hardware can write only a single format and density with the DC 6150 cartridges, so <code>/dev/rst8</code> doesn't change anything.

Remote File System Dumps Over the Local Area Network

You can dump file systems from a file server or standalone system over a network onto a tape mounted on another (remote) machine's tape drive. This is particularly useful if the local tape drive is down, though it, of course, requires that another tape drive be on the network.

If you have appropriate permissions, edit the remote machine's `/.rhosts`. Add the machine you dump from to the list of trusted machines. Thereafter, you can log in as superuser on your machine and do a dump to the remote machine. If you do not have the proper permissions for the remote machine, ask the machine's administrator to edit `/.rhosts` for you. The chapter *The SunOS Network Environment* explains `/.rhosts` in more detail.

The command you use for remote dumps is `rdump`. Its syntax is

```
# /usr/etc/rdump options remote_host:/dev/tape_dev /dev/file_system_to_dump
```

You use the `rdump` command to access a remote machine over the network. The arguments to the command are explained below:

- options* These options are similar to those used for dump. They include 0-9, for dump level, **c** to indicate a cartridge tape, **d bpi** for reel-to-reel tape density, **f** for dump file, and several others, as discussed in the `rdump` man page in the *SunOS Reference Manual*.
- remote_host:* The name of the remote machine with the tape drive you want to dump to. Be sure to include the colon (:) after the machine name, with no spaces before or after it.
- /dev/tape_dev* The device name of the tape drive on the remote host. Refer to the tables in the subsection, *Tapes and Equipment Used for Backup* for the proper device abbreviation for the remote host's tape drive.
- /dev/file_system_to_dump* The local disk partition containing the file system you want to dump. Refer to the table at the beginning of this subsection for the device abbreviation for your disk controller.

A Sample Remote Incremental Dump

This example assumes that you are backing up a file server with a Xylogics disk controller. The remote machine you want to access has a cartridge tape drive with a SCSI disk controller.

1. Log in to the file server as superuser.

NOTE You can use the same instructions for remote dumps to other types of configurations or to other file systems and dump levels. Simply substitute the appropriate arguments in the dump command line shown in Step 5.

2. If you are not certain whether the server is in the `/.rhosts` file of the machine with the tape drive, use `rsh` to execute a simple command such as:

```
# rsh cat /etc/motd
```

If you get a message like:

```
SunOS Release 4.1 (patches) #3: Mon Jun 19 16:05:15 PDT 1989
```

your server is in the remote machine's `/.rhosts` file. If `rsh` returns a

```
Permission denied.
```

message, your server is not in the remote machine's `/.rhosts` file. Contact the person responsible for the remote machine and have him or her add you to its `/.rhosts` file. You also must have an IP address entry in your `/etc/hosts` file for the remote machine.

3. Load the tape in the remote tape drive. (Or, have its administrator do this for you if the remote machine is not easily accessible from the server.)
4. Take the server down to single-user mode, as explained at the beginning of the subsection.
5. Type the following to incrementally dump the /home file system.

```
# /usr/etc/rdump 9uf remote_host:/dev/rmt8 /dev/rsd0h
```

6. Rewind the tape when the dump is complete by typing:

```
# rsh remote_host /usr/bin/mt offline
```

Then you can remove the tape from the remote machine. (Or, if it is at a distant location, have the administrator of the remote machine remove and label the tape.)

6.2. Restoring Files and File Systems

Restoring files after user deletion or crash may be the most significant function you have as system administrator. If you were dumping on a regular basis according to your planned dump strategy, you should be able to restore files and file systems to a reasonable state. If your tapes are properly labeled, you can easily determine which ones to use for the restore process after consulting the /etc/dumpdates file. (See below.) Then, after selecting the proper tape, you simply use the restore procedures in this section to recover your files.

Topics covered in this section include:

- Using the /etc/dumpdates file.
- restore command syntax.
- Recovering accidentally deleted files.
- Recovering broken file systems after a crash.

Using the /etc/dumpdates File

/etc/dumpdates keeps a record of each dump you take, provided that you have specified the **u** option of dump (as explained in the sections *Using the dump Command* and *Procedures for Dumping Your Configurations*). Here is a typical /etc/dumpdates file from a file server:

```

/dev/rxy0a      0 Fri Nov  6 07:54:38 1987
/dev/rxy0f      0 Sat Oct 10 07:53:44 1987
/dev/rxy0h      0 Sat Oct 10 07:56:57 1987
/dev/rxy0g      0 Sat May 23 08:02:34 1987
/dev/rxy0f      5 Fri Nov  6 07:55:20 1987
/dev/rxy0h      5 Fri Nov  6 07:58:08 1987
/dev/rxy0g      5 Fri May 29 09:03:07 1987
/dev/rxy0f      9 Thu Nov  5 07:15:51 1987
/dev/rxy0h      9 Thu Nov  5 07:18:04 1987
/dev/rxy0g      9 Thu Jun  4 09:21:02 1987

```

Each line in `/etc/dumpdates` gives information about the last dump performed at a particular level. The fields in the line include partition (file system) that was dumped, dump level, and dump date.

When you do an incremental dump, the dump command consults `/etc/dumpdates` to find the date of the most recent dump of the next lower level. Then it copies to tape all files that were updated since the date of that lower level dump. After the dump is complete, a new information line in `/etc/dumpdates`, describing the dump you just completed, replaces the information line for the previous dump at that level. On the date that you do a Level 0 dump, `/etc/dumpdates` will contain one information line for each file system at each level.

`/etc/dumpdates` cannot necessarily help you when you need to restore a file that might have been preserved during a particular backup. Rather, it is more important for you to develop and maintain a good dump schedule, and thoroughly understand your dump plan. Then you will find it easier to determine which dump tape contains the file you need.

On the other hand, `/etc/dumpdates` can help you in certain circumstances. You can use it to verify that dumps are taking place. This is particularly important if you think you are having equipment problems. If a dump can't complete due to equipment failure, a record of that dump will not appear in `/etc/dumpdates`. In addition, `/etc/dumpdates` is quite helpful if you ever need to restore an entire disk. The file will list the most recent dates and levels of dumps, so that you can determine which tapes you need to restore the entire file system.

Using the `restore` Command

The `restore` command copies files from tapes created by `dump` into the current working directory. You can use `restore` to reload an entire file system hierarchy from a Level 0 dump and incremental dumps that follow it, or to restore one or more single files from any dump tape.

The syntax of `restore` is:

```
# /usr/etc/restore key [names ..]
```

Here *key* refers to one or more of a number of keys available through `restore`. (*keys* differ from options in that they are not preceded by a dash.) These keys are

composed of one *function letter* and possibly one or more *function modifiers*. For a complete discussion of `restore`'s keys and other arguments, see the `restore(8)` man page in the *SunOS Reference Manual*.

The most frequently used function letters are:

- t** Verifies that the files you specified on the command line are on the tape. If `restore` finds the files, it displays their names on your screen. If you run `restore` with just the **t** function and no file name, `restore` lists all files on the tape (that is, provides you with a Table of Contents of the tape).
- i** Runs the interactive version of `restore`. The program runs an interactive environment that lets you select specific files to be restored.
- r** Tells `restore` to do a recursive restore; that is, copy everything on the tape.
- x** Restores only the files named on the command line.

The most frequently used modifiers are:

- v** Displays the names of each file as it is restored—the verbose option.
- f** Indicates that you are going to specify on the command line the tape device you want `restore` to read from.
- a** Takes the dump table-of-contents from the specified `archive-file` instead of from the dump tape. If the file that you request is in the table-of-contents, `restore` prompts you to mount the tape. If you only need contents information, for example when you use just the **t** option, you do not have to mount the dump tape.

The *[names ...]* argument is the name of the file, files, or directories whose files that you want to restore.

Restoring Individual Files

Frequently, a user will ask you to restore a file, files, or entire file system he or she has lost through carelessness, hardware failure, a faulty program, or by using a program improperly. Cases like this require you to be able to restore individual files or small groups of files.

Here are some actions to take before actually using `restore`, that make the restore process easier and more accurate.

1. Ask the user the date when the file was lost (in most cases, it's the present date), or the approximate date when the file last was in the state he or she wants to recover.
2. Refer to your backup plan to find the date of the last dump that would have the requested version of the file on it. Note that this is not necessarily the most recently backed up version of the file.

- Retrieve the tape containing the dump made on that date.

You should always be aware of the storage organization of backup tapes at your site, so that you can locate tapes that are months or years old. This is particularly important when you must restore a subdirectory containing many files worked on over a long period of time. In such cases, the needed versions of files are often not on a single dump tape.

- Typically, you restore files from the dump tape into the `/var/tmp` directory, though this is certainly not required. If you want to create a special directory to receive restored files, log in to your system as superuser and create this directory. You should almost never restore a file to its original directory.

Once you have taken these precautions, you are now ready to use `restore`.

Procedures for Restoring a File

In most cases, the following instructions apply to both server and standalone systems configurations, and to both 1/2 inch and 1/4 inch tape drives. The instructions include special steps for configuration-dependent or tape-size-dependent cases.

How to Verify a File, Then Restore It

- Log in as superuser and load the proper dump tape into the tape unit.
- Go to the directory where you want to copy the restored file:

```
# cd /var/tmp
```

If you don't want to restore into `/var/tmp`, substitute the name of the directory to receive the restored file.

- Type the following to verify that the file exists:

```
# /usr/etc/restore tf /dev/rmt8 file_name
```

The arguments to `/usr/etc/restore` are

t	The t function letter tells <code>restore</code> to verify whether the file, files, or directory given as the <i>file_name</i> argument are on the tape. f tells <code>restore</code> to find the file on the tape device named on the command line.
/dev/rmt8	The name of the tape device used. In this case it is a reel-to-reel tape drive using 6250 bpi tape. If you are using a different tape device, substitute the proper abbreviations for your device in this command.
<i>file_name</i>	The name of the file, files, or directory that you want to restore.

- Once you have found the file in the tape, restore the file by typing the following:

```
# /usr/etc/restore x /dev/rmt8 file_name
```

The **x** argument tells `restore` to copy the file, files, or directory `file_name`. If `file_name` is a directory, `restore` then recursively extracts the directory.

- Rewind the tape and take it offline by typing:

```
# mt -f /dev/rmt8 offline
```

It is suggested that you let the user change the restored file to whatever name he or she chooses and move the file to its desired location.

How to Interactively Restore Files

The next procedure uses the interactive version of `restore`.

- Log in as superuser and load the proper dump tape in the tape unit.
- Go to the directory where you want to copy the restored file:

```
# cd /var/tmp
```

If you don't want to restore into `/var/tmp`, substitute the name of the directory to receive the restored file.

- Type the following:

```
# /usr/etc/restore ivf /dev/rst8
```

Here the **i** function letter invokes the interactive version of `restore`. `/dev/rst8` is the name of the tape device you are going to use. In this example, it is a nine-track cartridge tape device. If your system has a different tape device, substitute the proper device abbreviation here. Remember, you do not have to specify the blocking size for a tape cartridge device, since this is already written on the tape label.

The system invokes `restore` in interactive mode. You will see the following on your screen:

```
Verify tape and initialize maps
Tape block size is 126
Dump   date: Tue Aug 18 09:06:43 1987
Dumped from: Fri Aug 14 08:25:10 1987
Level 0 dump of /usr on pilgrim:/dev/sd0g
Label:none
Extract directories from tape
Initialize symbol table.
restore >
```

You type instructions after the `restore >` prompt shown above. Interactive `restore` understands a limited set of commands.

4. Display a directory using the `ls` command to verify that it has the files you want to restore:

```
restore > ls
```

- 5a. If the file is in another directory, change to that directory using the `cd` command:

```
restore > cd directory_name
```

- 5b. After finding your file, add it to the list of files you want to restore:

```
restore > add file_name
```

You can also give `add` the name of directories you want to restore. Also, you need to consider at this time that all the files in a directory may not be on a single dump tape. If this is the case, you may need to restore from multiple dump levels to get all the files that you need.

6. Restore the list of files by using the `extract` command:

```
restore > extract
```

7. Specify "volume 1" when asked for a volume number. If you are doing a multi-volume restore, you should start by specifying the last tape first. This will speed up the restore process.
8. Type `q` to leave interactive `restore`.
9. Complete Steps 4 and 5 of the previous procedure.

Restoring Files Over a Local Area Network .

You can restore file systems from a remote tape drive to a file server or standalone system on a local area network. In order to do this, the name of the server or standalone system from which you are issuing the `restore` commands must be in the `.rhosts` file on the machine with the remote tape drive.

Use the `rrestore` command with the following syntax:

```
# /usr/etc/rrestore keys remote_host:/dev/tape_dev file_name
```

`rrestore` is a version of `restore` used specifically for accessing a remote machine over the network. The arguments to the command are explained below:

- | | |
|----------------------|---|
| <i>keys</i> | These keys are the same as those used for <code>restore</code> . Refer back to the subsection, <i>Using the restore Command</i> , for more information. |
| <i>remote_host:</i> | The name of the remote machine with the tape drive you want to use. Be sure to include the colon (:) after the machine name, with no spaces before or after it. |
| <i>/dev/tape_dev</i> | The device name of the tape drive on the remote host. |
| <i>file_name</i> | The name of the file or files you want to restore. |

A Sample Remote Restore

This example assumes that you are using an NFS server and accessing a remote machine with a cartridge tape drive attached to a SCSI disk controller.

1. Log in to the NFS server as superuser.
2. If you are not certain whether the server is in the `.rhosts` file of the remote machine, use `rsh` to execute a simple command such as:

```
# rsh remote_host cat /etc/motd
```

Where *remote_host* is the name of the remote machine. If you get a message like

```
SunOS Release 4.1 (patches) #3: Mon Jun 19 16:05:15 PDT 1989
```

your server is in the remote machine's `.rhosts` file. If `rsh` returns a

```
Permission denied.
```

message, your server is not in the remote machine's `.rhosts` file. Contact the person responsible for the remote machine and have them add you to its `.rhosts` file. Also, add the remote machine's IP address to your `/etc/hosts` file.

3. Load the tape into the remote tape drive. (Or, have its administrator do this for you if the remote machine is not easily accessible from the server.)
4. Go to the directory where you want to copy the file.

```
# cd /var/tmp
```

If you do not want to copy the files to `/var/tmp`, substitute the name of the directory you want to use in this command.

5. Type the following:

```
# /usr/etc/rrestore xf remote_host:/dev/rst8 file_name
```

6. Rewind the tape when you have restored the files by typing:

```
# rsh remote_host /usr/bin/mt -f /dev/rst8 offline
```

where *remote_host* is the name of the remote machine. Then you can remove the tape from the remote machine. (Or, if it is at a distant location, have the administrator of the remote machine remove the tape.)

Restoring an Entire File System

Occasionally a file system becomes so damaged that you have to restore it entirely. Typically, this is due to a disk head crash. (If this is the case, you probably need to replace the disk.) Fully restoring a file system such as `/home` can take a lot of time. If you have faithfully backed up your file systems, you can restore them to their state as of the previous working day—or at least up to the last incremental dump, if your site does not require daily incremental dumps.

The Strategy to Restore a Whole File System

This overview provides a good strategy for restoring whole file systems:

- a. Replace any broken hardware.
- b. Unmount the file system.
- c. Check that the file system is indeed unmounted.
- d. Re-make the file system on the disk.
- e. Check that the file system was properly re-made.
- f. Mount the file system on a temporary mount point.
- g. Restore the contents of the tape.
- h. Unmount the file system from its temporary mount point.
- i. Check the file system for inconsistencies.
- j. Mount the file system at its permanent mount point.

When restoring the file system, you use three commands that have not been previously covered in this manual:

```
umount
newfs
mount
```

They are described in the next subsections.

The umount Command

Before you can restore the file system, you must first unmount it. To unmount a file system, use the `umount` command as follows:

```
# /usr/etc/umount /dev/disk_type_partition
```

where *disk_type_partition* is the partition containing the file system you want to unmount. Now verify that it is unmounted by entering the `mount` command:

```
# mount
```

`mount` lists all the currently mounted file systems. Here is a sample of `mount` from displays on an NFS file server:

```
/dev/xy0a / 4.2 rw,nosuid 1 1
/dev/xy0g /usr 4.2 rw 1 2
/dev/xy0e /export/swap 4.2 rw 1 4
/dev/xy0d /export 4.2 rw,nosuid 1 3
/dev/xy2h /home 4.2 rw 1 3
```

If the file system is NOT listed in the display, then you can be pretty sure that it is not mounted. If the file system does remain mounted, halt the machine and bring it up in single user mode.

```
# /usr/etc/halt
> b -s
```

Refer to the chapter *The Sun Network File System Service* and `umount(8)` in the *SunOS Reference Manual* for more information.

The newfs Command

Next, you need to run the `newfs` command. `newfs` creates a new file system on the specified disk partition. Its basic syntax is

```
# /usr/etc/newfs options /dev/rdisk_type_partition
```

where */dev/rdisk_type_partition* is the disk controller name and disk partition where you want to build the new file system.

`newfs` has a fair amount of options that are not covered in this manual. Moreover, it is a variation of the more involved `mkfs` command, which you don't

need to use when restoring hard partitions. Refer to `newfs(8)` and `mkfs(8)` in the *SunOS Reference Manual* for more information.

Mounting a Local File System

SunOS Release 4.1 provides several types of mounts. One type of mount is NFS, described in Part II. Another mount type, RFS, pertains only to RFS domains, also explained in Part II. The third type of mount is a 4.2, or *UFS mount*, which refers to mounting a file system from a local disk. This is the type of mount used during a restore procedure.

When a file system is mounted from a local disk, the file system is attached to a location in your local directory hierarchy so that users can access the contents of the file system. Because this is a local mount, the file system is an entire partition of your disk. `/etc/mntab` lists local mounts as 4.2 mounts, because the mount performed is the traditional mount provided by the BSD 4.2 operating system. The `/etc/mntab` file for a file server lists as 4.2 mounts all file systems that the server mounts from its local disk. Any file systems the server might receive from another machine on the network would be listed as NFS or RFS mounts.

The syntax you use to perform a local mount is:

```
# /usr/etc/mount /dev/disk_type_partition /mnt
```

where `disk_type_partition` is the disk controller device abbreviation and the partition letter for the file system you want to mount. `/mnt` is the directory or *mount point* where you want to attach the file system. If required, you can use a different directory than `/mnt`, but this directory must be an empty directory or *mount point*, or the contents of the directory will remain inaccessible for as long as the mount is in effect. The chapter *The Sun Network File System Service* describes mount points in more detail.

When you want to remove a file system, you use the `umount` command as shown below:

```
# /usr/etc/umount /dev/disk_type_partition
```

where `disk_type_partition` is the partition containing the file system you want to unmount.

Determining Which Tapes to Use

Before you perform the restore procedure, you need to determine which dump tapes to use. When restoring an entire file system, you always need the most recent Level 0 dump tape. You also need the most recent incremental dump tapes made at each of the higher levels.

For example, if you just make Level 0 and Level 9 dumps, you only need the last Level 9 dump made, since it picked up all changes made since Level 0. If you use a plan similar to the ones illustrated in *Planning a Dump Strategy*, you will need the most recent Level 0, Level 5, and Level 9 dumps.

Procedures for Restoring an Entire File System

Note: Do not use these instructions for restoring an entire root (/) or /usr file system on either a standalone system or server. Instead, use the instructions in the subsection, *Restoring a Damaged / or /usr File System*.

The next procedure shows how to restore the /home file system on a file server. You can use this procedure for restoring other file systems, including the /export/root file system on a server, and file systems you may have created for your site.

The procedure shows this type of restore operation by using an example. It assumes that you are restoring the /home file system on partition *h* of a file server with a Xylogics 450 hard disk controller and a reel-to-reel tape drive using 6250 bpi tape. If the device abbreviations in the examples don't apply to your system, substitute the proper abbreviations, as shown in the device tables in *Backing Up File Systems*.

This procedure assumes that a serious software or hardware problem resulted in the loss of the /home file system. It also assumes that you have corrected the problem, have determined the tapes you need to use for the restore, and are now ready to restore the file system from the tapes.

1. Log in to the system as superuser.
2. Unmount the affected file system.

```
# /usr/etc/umount /dev/xy0h
```

Check to be sure the file system is not mounted:

```
# /usr/etc/mount
```

3. Recreate the damaged file system by using newfs:

```
# /usr/etc/newfs /dev/rxy0h
```

where /dev/rxy0h is the partition where you want newfs to build the new file system. Note that you address the raw device with *r*.

4. Check the file system by typing:

```
# /usr/etc/fsck /dev/rxy0h
```

Remember to specify the raw device.

Be cautious about running fsck on mounted file systems. It is safer to run it in single-user mode or on unmounted file systems. At this point, any problems that fsck reports indicate that something is seriously wrong.

5. Mount the file system on an existing directory before reloading. Typically, you use the /mnt directory.

```
# /usr/etc/mount /dev/xy0h /mnt
```

Note: Here you specify the block device, not the raw device.

6. Load the Level 0 dump tape; go to the `/mnt` directory; and restore the file system.

```
# cd /mnt
# restore rvf /dev/rmt8
```

7. Rewind the Level 0 tape, remove it, then load the next lowest level dump. Always restore tapes starting with the lowest level and continuing until you reach the highest level.
8. Use the same version of `restore` to copy the incremental dump tape. If you have more tapes to load, repeat Steps 6 and 7 for each tape.
9. Remove the file `restoresymtable` from `/mnt` (or your current directory).

```
# rm restoresymtable
```

`restore` creates this directory during its operation.

10. Go to another file system. (You can't unmount a file system while you are still in it.) Then, unmount the file system and check it with `fsck`.

```
# cd /
# /usr/etc/umount /dev/xy0h
# /usr/etc/fsck /dev/rxy0h
```

11. Because `restore` runs in user mode, it has no control over inode allocation; this means that `restore` repositions the files, although it does not change their contents. Therefore it is necessary to do a full dump of the newly restored file system:

```
# /usr/etc/dump 0uf /dev/rmt8 /dev/rxy0h
```

This dump sets up a new set of directories reflecting the new file positions, so that later incremental dumps will be correct.

Restoring a Damaged / or /usr File System

Restoring a damaged root or `/usr` file system from a dump tape is unlike other types of restore operations, because the programs you need to run are in the damaged file system. Therefore, you first need to load and boot the mini-root file system from the boot tape, and reload the file systems from there.

Procedures for Restoring `root`

The procedures below are appropriate for file servers and standalone systems. The example is for a standalone system using a SCSI disk controller and a SCSI tape cartridge unit with nine-track tape. If the device abbreviations in the example do not apply to your configuration, remember to substitute the appropriate abbreviations when you need to do these procedures.

1. Locate and fix any bad blocks or other hardware problems. See the *Maintaining Disks* chapter for more information on fixing bad blocks.
2. Load and boot the mini-root file system from the boot tape:

```
>b st(0,0,0)
Boot: st(0,0,2)
From: st(0,0,3)
To: sd(0,0,1)
.
.
Standalone copy takes place
.
.
Boot: sd(0,0,1)vmunix -asw
```

If you have disk problems, boot `format` from tape and fix the disk. Be sure to check that the disk label is good by using `format`'s `label` option. Refer to the section *Relabeling a Corrupted Disk* in the chapter *Maintaining Disks* for more details.

3. Use the `MAKEDEV` command to install the entries for the tape unit and disk type onto the mini-root file system. (The `MAKEDEV` command is fully explained in the chapter *Adding Hardware to Your System*).

```
# cd /dev
# MAKEDEV st0
# MAKEDEV sd0
```

4. Use the `newfs` command to recreate the root file system:

```
# /etc/newfs /dev/rsd0a
```

Use `r` for raw device.

5. Enter

```
# newfs -n
```

to recreate the superblock numbers.

6. Check the root file system:

```
# /etc/fsck /dev/rsd0a
```

7. Mount the file system so it can be restored. The next example assumes the `/mnt` directory does not exist; if it does exist, do not run the `mkdir` command shown below.

```
# mkdir /mnt
# /etc/mount /dev/sd0a /mnt
```

8. Change to the `mnt` directory and restore the level 0 dump tape.

```
# cd /mnt
# /etc/restore rvf /dev/rst8
```

9. Continue to restore incremental dumps in the order of lowest level number first and working up to the most recent, highest level tape last.
10. Remove the `restoresymtable` file that `restore` creates in the working directory during its operation.

```
# rm /mnt/restoresymtable
```

11. You must install a new boot block after restoring the root partition. Use the `installboot` command below to create the boot block.

```
# cd /usr/mdec
# installboot /mnt/boot bootsd /dev/rsd0a
```

12. Unmount the file system and check it again with `fsck`.

```
# cd /
# /etc/umount /dev/sd0a
# /etc/fsck /dev/rsd0a
```

Notice that you specify `r` to have `fsck` check the raw device.

13. Do a full dump (Level 0) of the newly restored root file system.

```
# /etc/dump 0ucf /dev/rst8 /dev/rsd0a
```

Restoring the `/usr` File System

The instructions for recovering `/usr` are similar to recovering `/`, but vary from site to site. If you have a file server and left `/usr` in the same state as it was when installed from the tape, you only need to use a Level 0 dump of `/usr` to restore the file system. However, if you have moved any site-specific software into `/usr` on a file server, particularly into the `/usr/local` directory, the `/usr` file system may be more volatile. Thus, you may have taken incremental dumps of `/usr` that you might want to use in addition to the last Level 0 dump. Therefore you not only have to restore a Level 0 dump of `/usr`, you also have to restore the latest version of each higher level dump taken since the Level 0 dump.

The next instructions refer to the subsection *Procedures for Restoring root*, and explain only the steps that are unique to restoring `/usr`.

1. Perform steps 1-7 as shown in *Procedures for Restoring root*. Remember to substitute the `g` partition for `/usr` (as opposed to the `a` partition for `/`) when specifying disk controller and tape device names.
2. For Step 8 on a file server, you probably do not need to do a higher level dump than 0, since there are no user files in `/usr` initially, and you have probably mounted it read-only. However, if you are on a standalone system, this is where you have to load incremental dump tapes.
3. Skip Step 9 in *Procedures for Restoring root*. You only have to create a new boot block when you are restoring `/`.
4. Perform Steps 10, 11, and 12.

6.3. Monitoring File System Usage

Besides backing up and restoring files, the other file system maintenance operation you must perform on a regular basis is checking file system usage. This section describes how you can keep track of file system usage, including:

- Determining the extent of file system usage.
- Determining how much disk space is being used.
- Reducing overloaded file systems.

The next information applies to all Sun machines: diskless and dataless clients, standalone systems, time-sharing systems, and file servers.

Checking File System Usage

As you did for backup, you should devise a schedule for checking file system usage. How often you do this should depend on your configuration type, how many people use it, and how file-intensive the work they do is. For example, if your Sun workstation mainly runs large FORTRAN programs that perform mathematical calculations, this is not as file-intensive as using it to create large application programs or documents.

The SunOS operating system provides several commands that help you determine file system usage:

```
ls
du
quot
find
```

These four commands allow you to quickly find large files. If disk storage space is short, you can move unused large files to a less expensive medium, like tape.

ls and File System Usage

The familiar `ls` directory listing command has several options that you can use to determine file system usage.

- `ls` with the `s` option provides the size in kilobytes of the files in a directory.

- The command

```
% ls -R directory_name
```

recursively lists all subdirectories in the given directory.

- To find the largest files in the current directory, type:

```
% ls -s | sort -nr | more
```

- To find out which files were most recently written, use `ls -t`. It lists files in order of most recently created, or altered, first.

The du Command

`du` gives the number of kilobytes contained in all files and, recursively, directories within each specified directory or file named.

The syntax for `du` is:

```
% du [-s|a] filename
```

The options are:

- s** Displays the total for each of the specified filenames.
- a** Generates an entry for each file.

In the absence of options, entries are generated only for each directory.

The following example shows what appears on your screen after you enter the `du` command.

```
% du
39    ./Release/admin
357   ./Release/bugs
628   ./Release/ssp
7273  ./Release
```

The left column shows the number of kilobytes found in files of the directories on the right. The last line of the `du` command reports the total size of the hierarchy. You can request only the total size with the `s` option as follows:

```
% du -s .
7273 .
```

See the `du(1)` man page for a complete description.

The quot Command

To use `quot`, you must become superuser. This command gives the number of blocks (1024 bytes) currently owned by each user in the named file system.

The syntax for `quot` is:

```
# quot [-acf] filesystem
```

Some options are:

- a** Generates a report for all mounted file systems.
- c** Displays three columns giving file size in blocks, the number of files of that size, and the cumulative total of blocks in that size or smaller file.
- f** Displays a count of the number of files and space each user owns.

When using the **c** and **f** options, you must specify the file system. Use the `quot` command as follows to see a report of all the mounted file systems:

```
# quot -a
/dev/rsd0a (/):
 3819  root
   97  cat
   49  uucp
   17  bin
/dev/rsd0e (/export):
  13  root
/dev/rsd0g (/usr):
34518  root
 2182  bin
  638  uucp
/dev/rsd0h (/home):
  10  root
   2  cat
```

See the `quot(8)` man page for a complete description.

The find Command

The `find` command, searches for files that you specify. The syntax for the `find` command is as follows:

```
% find pathname-list expression
```

pathname-list Is where the `find` begins its search. You can list one or more pathnames separated by a space. This is helpful because you can search several different places in the directory tree without searching the whole tree.

expression Is one or more operators (see below) that you use to describe which file(s) you are looking for.

In the operator descriptions below, *n* is a decimal integer where *+n* means more than *n*, *-n* means less than *n*, and *n* means exactly *n*.

The following are some of the operators you use to create a logical *expression*.

name filename	Use to find a <i>filename</i> argument that matches the current filename.
user uname	Use to find a file that belongs to the user <i>uname</i> . If <i>uname</i> is numeric, and does not appear as a login name in the <i>/etc/passwd</i> database, it is taken as a user ID.
nouser	Use to find a file that belongs to a user not in the <i>/etc/passwd</i> database.
size n	Use to find a file that is <i>n</i> blocks long. (512 bytes per block.) If you follow <i>n</i> with a c , the size is in characters.
print	Use to print the files's pathname, if found, to standard output. Unless you specify print , <i>find</i> performs silently.
ctime n	Use to find a file that was changed in <i>n</i> days.

In the following example, *find* finds all the files owned by user *cat* over 800 blocks and prints those files to the screen.

```
% find /usr/cat -user cat -size +800 -print
```

You can also use the *find* command to find all files that are older than a certain date and do not belong to anyone in the */etc/passwd* file. For example

```
% find /usr/sam -nouser -ctime 180
```

finds all the files that were changed in less than 180 days.

You can also concatenate various operators. For example,

```
% find /home -user cat -o -user jim -print
```

prints the names of all the files under the */home* that belong to users *cat* or *jim*.

```
% find /home -user cat -a -name '*.tmp' -print
```

finds all the files under */home* that belong to user *cat* and have the suffix *.tmp* in the name. Note that you can use shell metacharacters, but you must escape them.

```
% find /home -name core -o -name '*.tmp' -o -name dog -exec rm -f {} ;
```

finds all files under the */home* directory named *core*, or *dog*, or have a *.tmp* suffix, and removes them.

See the *find(1)* man page for a complete description.

Checking Disk Usage

The `df` command displays information about specific disk partitions that you can use to determine disk usage. This information includes file space occupied by each file system, space used and space available, and how much of the file system's total capacity has been used. When `df` gives the percentage of the capacity used on the disk, the number indicates the optimum capacity. When a file system is 90 percent full, `df` claims that it is 100 percent full. If the file system becomes any fuller, file system access slows down. In fact, when a file system becomes 90 percent full, only the superuser may continue creating or increasing the size of the files. Ordinary users get an error if they attempt to go beyond this limit.

Here is a display generated by the `df` command on an NFS file server:

File system	kbytes	used	avail	capacity	Mounted on
/dev/xy0a	7735	6200	761	89%	/
/dev/xy0g	232507	74366	134890	36%	/usr
/dev/xy2h	548367	486737	6793	90%	/home
/dev/xy3d	76111	28728	39771	42%	/export
/dev/xy0e	57335	42274	9327	82%	/export/swap

Note that this file server has three disks mounted and has divided file systems among them.

Making Room on File Systems

You should always monitor disk usage so that you are not caught in a situation where a file system has run out of space. Unfortunately, on heavily-used machines this is not always possible, resulting in overloaded file systems.

This subsection shows how to free up space on an overloaded file system. The instructions apply to all Sun computers. Remember that even if you administer a diskless client, you still need to check usage of the `/home` file system on your server, and, if necessary, delete unneeded files.

You know file system space has been exceeded when you get the following message on your workstation console:

```
file system full
```

Before trying anything else, suspend or kill the currently executing programs and try to make more room on the affected file system.

For systems that are filling up, or that become full suddenly, delete some files or move some files to another file system. Try the following:

1. Check directories for files named `core`, and delete them. (When certain programs crash, they leave a `core` file.)
2. If you enable crash dumps, you should check the `/var/crash` directory. At the next reboot after a crash, `savecore`, a program which runs from `/etc/rc.local`, puts a copy of the core dump in `/var/crash`. This copy will stay in `/var/crash` until you remove it. If a second crash happens, `savecore` will put a second core dump in `/var/crash`, making

the contents of this file system even greater.

As distributed, the section of `/etc/rc.local` that creates the crash dump file is commented out. Should you wish to preserve crash dumps without using up too much disk space, create a file called `minfree` in `/var/crash`. `savecore` reads from this file. If the file system contains less free space in kilobytes than the ASCII number contained in `minfree`, `savecore` will not save the core file.

3. Check the directory `/var/adm`. It contains accounting files that increase in size as you use your system.

You can disable login accounting by removing the file `/var/adm/wtmp`. You can re-enable accounting by creating the file in question with length zero. To do this enter:

```
# touch /var/admin/account
```

You can disable process execution accounting by typing

```
# /usr/lib/acct/turnacct off
```

You can then re-enable accounting by typing

```
# /usr/lib/acct/turnacct on
```

The files `/var/adm/messages.x` preserves system messages that the system displays in your console window or screen. These files are useful for debugging system problems, but they can grow very large. Check the `messages` files periodically. When one grows enormous and contains messages that are at least a month old, edit the file and delete its contents. Do not delete the file itself. The easiest way to accomplish this is to enter:

```
# cp /dev/null /var/admin/messages.x
```

4. Look for obsolete files in `/tmp` and `/var/tmp`. Be careful not to delete currently active temporary files from `/tmp`, such as those filenames starting with `Ex` or `Rx`, which the editors use. Also look for obsolete files in the subdirectories of `/var/spool`, such as `/var/spool/mail` and `/var/spool/uucppublic`.

6.4. The Disk Quota System

Disk space is always a limited resource. The disk quota system provides a mechanism to control how much disk space each user uses. Typically, the quota system is used for a file server, but you can also use it for a standalone systems. You should enable the quota system if your site:

- Has limited amounts of disk space

- Requires tight security
- Has high disk usage, as is the case in many universities, for example

If these conditions do not apply to your site, you probably do not want to use the quota system.

If you are the administrator of a diskless or dataless client upon which the network has imposed quotas, refer to the section, *Quotas and Clients*, at the end of this section.

The quota system is an optional part of the kernel that you can specify when the system is configured. You can set quotas for each user on any or all file systems. The quota system warns users when they exceed their allotted limit, but allows some extra space for current work. A user who remains over quota longer than a specified time-limit will get a fatal over-quota condition.

Setting Up the Quota System

You must perform these steps to set up the disk quota system.

1. Become superuser on your file server.
2. Verify that the system is configured to include the disk quota subsystem.

```
# cd /usr/kvm/sys/sun[3,3x,4,4c]/conf
```

3. Edit the kernel configuration file, as described in the instructions in the chapter *Reconfiguring the System Kernel*. If it is not already there, add the line:

```
options    QUOTA
```

Note that if you add or enable the quota option, ensure that the line

```
options    UFS
```

is also enabled in the kernel configuration file.

4. Rebuild the kernel, as described in *Reconfiguring the System Kernel*.
5. Decide which file systems need to have quotas imposed. Usually, only the `/home` file system on the file server and `/usr` on a standalone system need quotas. (If you have created other file systems that have users' files on them, you may also want to impose quotas on them.) You may also want to impose quotas on `/usr` on a server. If possible, `/tmp` should be free of quotas. Work out your plan before you begin the actual allocation. After you decide which file systems should have quota restrictions, allocate available space according to your plan.
6. Go to the top directory of the file system that will have quota restrictions.
7. Create a file called `quotas`. Make sure that the new `quotas` files are owned by root. If necessary, use the `chown` command to change their ownership.

The `edquota` command actually sets quotas for specified users. `edquota` is a quota editor that allows you to set limits for one or more users at a time. Use the `p` option of `edquota` to duplicate the quotas of a prototypical user for a list of actual users. This is helpful when many or all users are given the same limits. The `t` option edits the over-quota time-limit for the specified users. You can impose any combination of hard and soft limits for each user. A limit set to zero is disabled. If all limits for a user are disabled, the entire quota for that user is also disabled, and the system will not keep track of his or her usage. For details see the man page `edquota(8)`.

Turning on the Quota System

Once quotas are set and ready to operate, you must turn them on before they can start working. The `quotaon` command tells the system to start enforcing the quotas set with `edquota`. `quotaon` enables disk quotas for one or more mounted file systems.

The `quotaoff` command disables quotas, as fully described in the *SunOS Reference Manual*. However, when you are about to unmount a file system through `umount`, quotas are disabled just before the specified file system is unmounted. This guarantees that the quota system will not be disabled if the unmount fails because the file system is busy.

The file `quotas` holds information about disk quota use and limits. This is the file you created at the top level of the file system where the quotas are imposed. For example, the `quotas` file for the `/home` file system should be located directly under `/home`. The name `quotas` is not known to the system in any way. However, several user level utilities depend on it, so choosing any other name would cause problems.

The information in `quotas` is a list that user ID indexes. It contains one entry for each user on the machine, whether or not the user has a quota on this file system. If the user ID space is sparse, the file may contain holes that would be lost by copying it. Therefore, avoid copying the file.

Administering the Quota System

The Sun disk quota system is based on the 4.2BSD quota system. The SunOS quota system limits the amount of time a user can be over quota. You can adjust that time-limit on a per-file-system basis using `edquota`. You can run quota administration commands only on mounted file systems. The commands will work whether or not the quota system is disabled.

You should periodically check the records retained in the quota file for consistency with the actual number of blocks and files allocated to a user. You certainly should do this after each reboot, and when quotas are first enabled for a file system. Do this using `quotacheck`.

When you are superuser, you can use `quota` to examine the use and quotas for an individual, and `repquota` to check the usages and limits for all users on a file system.

You can increase or decrease a users quota limits at any time. Thus, user's who exceed their limits can ask you to increase them.

When you remove a user from a system, you should use `edquota` to set the removed user's limits to zero. Then, when the user's login is removed, the `quotas` file will not have an entry for it.

The User's View of Disk Quotas

The `quota` command gives relevant information to a user upon whom you have imposed quotas. It also reports on an individual's current usage. You can impose two kinds of limitations on a user. You can set a quota on the amount of disk space a user can fill up, or you can limit the number of files (inodes) the user can own. Typically, you impose both of these limitations.

The disk quota system has soft limits and hard limits. The soft limit is the number of 1K blocks (or files) that the user is expected to remain below. Each time the user exceeds this limit, a warning is displayed and a time limit set. If the user remains above quota and the time limit expires, the system now acts as if the hard limit has been reached, and no longer allocates resources to the user. The only way to reset this condition is for the user to reduce usage below quota. You may set the over quota time limits for each file system restricted by quotas.

The user cannot exceed the hard limit. If usage reaches this number, the system displays an error message in response to any further requests for space, or attempts to create a file. The first time this occurs, the user's machine receives a message. Only one message is displayed for each time the hard limit is reached. Space occupied must be reduced below the limit.

What To Do When Quota Limit Is Reached

To recover from exceeding the hard quota limit, the user must do the following:

1. Abort the process or processes in progress on the file system that has reached its limit.
2. Remove enough files to bring the limit back below quota.
3. Retry the failed program.

The case is different when the user is in an editor and cannot write a file to a disk because it exceeds the quota. Most likely, the write attempt that generated the quota exceeded message also truncated the file in the editor. If the user aborts the editor in these circumstances, he or she will not only lose recent changes, but probably some, or all, of the file proper, as well.

There are several safe exits from this situation. The user may escape from the file to the shell, using the appropriate escape command, examine file space, then remove surplus files. However, it is probably quicker to suspend the editor session while removing files.

For example, if the user types `CTRL-Z` while in an editor, he or she is returned to the shell for as many commands as needed to remove files and get below quota limit. To continue the editor session, the user types `fg` (for foreground) to the shell prompt, and the suspended editor will restart. A third possibility is to write the file to some other file system (perhaps to a file on `/tmp` where your quota has not been exceeded.) After rectifying the quota problem, the user can move the file back to the file system where it belongs.

Quotas and Clients

Quotas on remotely mounted file systems work much the same as quotas on locally mounted file systems. The important difference is that the user will not receive a warnings upon reaching the soft limit. On a network client, use `quota` to find out the state of your quota allocation. Hard limit errors are treated like other hard errors in the network— for example exceeding actual disk capacity. When users exceed the hard limit, they get return code errors.

Administering Security

This chapter explains the basic features provided for security administration by SunOS Release 4.1, including these areas.

- An overview of security administration
- Preventing unauthorized access to your local machine by setting up the `/etc/passwd` and `/etc/group` files
- Protecting files and directories through discretionary access control
- Ensuring the integrity of running processes
- Protecting your system and network from computer viruses, Trojan horses, and other would-be invaders
- Suggested security procedures that you can set up for your site

The text assumes you are familiar with the security concepts introduced in the *SunOS User's Guide: Getting Started* and in the *SunOS User's Guide: Doing More*.

7.1. Security on Your Machine--an Overview

Keeping the computer system's information secure is a paramount system administration responsibility. However, different sites require different degrees of security. Sites that handle particularly-sensitive information require stringent security measures; others may prefer an open environment with little or no security measures in place. Before implementing security measures for your machine or network, carefully consider the relative confidentiality of the information maintained. Moreover, if you administer a file server or time-shared system, you must also poll the other users to determine their security requirements. Thus armed, you can best determine which security features described in this chapter and the remainder of *System and Network Administration* to implement in your environment.

SunOS Release 4.1 provides the following security features:

- Standard UNIX security features that you should implement for your local machine. Commands such as `passwd`, `chmod`, and `umask` are common to all UNIX-based systems. The SunOS Users' Guides introduce them. This chapter touches on them briefly and introduces more security-related commands and files common to UNIX-based systems.

- Network security features, such as the `.rhosts` and `/etc/hosts.equiv` files, which enable you to control remote logins and commands on your machine. See the chapter *The SunOS Network Environment* for more information.
- Basic NFS security measures that protect the integrity of files shared through the NFS service, as described in the *The Sun Network File System Service* chapter.
- RFS security features that protect the integrity of files shared through the RFS service, as described in the *The Remote File Sharing Service* chapter.
- Security-related databases that you can set up through the NIS and DNS name services. See chapters *The Network Information Service* and *Administering Domain Name Service* respectively.
- Secure NFS features that provide an option to mount secure file systems, requiring DES authentication of user and host. *The Sun Network File System Service* explains how to set up secure file systems and DES authentication.
- An install-time option to run systems at a moderately high level of security, patterned after the widely-accepted C2 classification. The chapter *Administering C2 Security* explains how to implement C2-style security on Sun computers.

Protecting System Integrity through Security Barriers

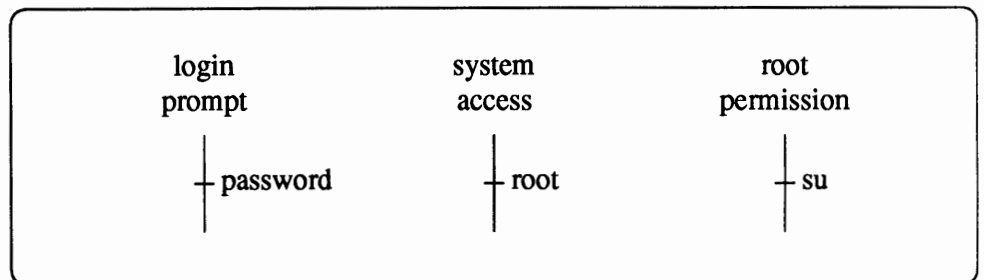
The primary goal of computer security is to protect data privacy and integrity. That is, data should not be read by those not authorized to read it, nor written by those not authorized to write it. Historically, the most common way of attacking security on UNIX-based systems has been to gain `root` permission, at which point all files on the system are readable and writable. Other less-pervasive attacks on UNIX security involve forging the credentials of a particular user, at which point that user's files are readable and writable. Needless to say, the former method of attack is more pernicious, but the latter method also compromises system security.

SunOS Release 4.1 provides a set of barriers for the safekeeping of data. You could think of these barriers as a set of hurdles that an attacker must jump over before reaching the goal of gaining unauthorized access to data. There are two types of local security barriers: system security and data security. Network security combines both system and data security barriers.

System Security Barriers

Most important to your local system are the system security barriers, as shown in Figure 7-1:

Figure 7-1 System Security Barriers



The first barrier an intruder must cross is the login screen or prompt. To cross this barrier, a would-be user must supply a user name and corresponding password known by the local machine.

The second and third barriers protect the system files and programs. To cross this barrier, a would-be superuser must supply the root user name and its correct password. Most sensitive parts of the system, such as the kernel, memory files, and device drivers, can only be accessed by the superuser. When you log in to the system under your user name, you can only access information for which you are granted permission. You cannot run important system commands, such as `dump`, to back up files, or `config`, to reconfigure the kernel, without first becoming the superuser. You can become superuser by logging in as root from the login prompt or by issuing the `su` command. *The System Administrator's Role* explains how to use both methods.

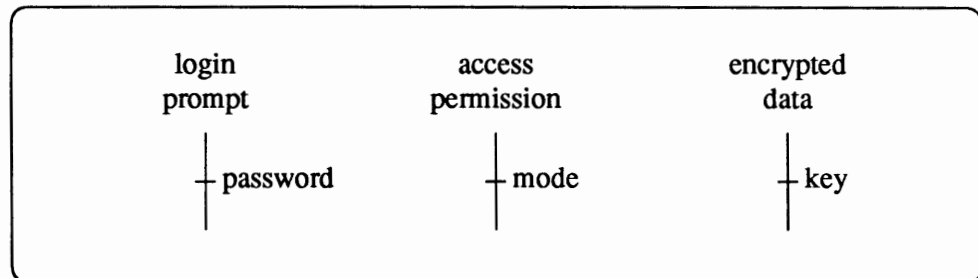
Note: You do not have to run C2 security to make your console secure. Any system administrator can do this, but it is the default with C2

If your system is configured as C2 secure, or the `console` entry in `/etc/ttytab` is not marked as physically secure, booting single-user requires the root password. This prevents unauthorized users from booting single user in order to make system modifications or to change the root password. Furthermore, systems where the `console` entry in `/etc/ttytab` is not marked as physically secure do not allow root logins. If you administer this type of system, you must log in with your user name, then use `su` to gain superuser access. This assures that when security-related actions are audited, the user's login name also gets recorded. (See the chapter *Adding Hardware to Your System* for more information about the `ttytab` file.)

Data Security Barriers

System security is essential for preserving the privacy of data. Most users, however, are not system administrators, and are more concerned with the privacy of their own data than with maintaining system security. Figure 7-2 shows the barriers that guard the privacy of user data:

Figure 7-2 Data Security Barriers



The first barrier that provides data security is, once again, the login prompt. The first task every Sun computer user performs daily is to log in and supply the password matching his or her user name.

The second data security barrier is the access permissions assigned to files and directories. (These all-important UNIX concepts are introduced in the SunOS User Guide series.) Once logged in, you may access any files for which you are granted permission. On secure systems, you are the only user who may read and write your files. However, you can give away read, write, and execute permission to members of a group, or to anybody at all.

For an added measure of privacy, you can employ the third data security barrier: encrypting your files with the `crypt` or `des` programs. You can use `crypt` or `des` if you have installed the encryption kit optional software package available within the United States. See `crypt(1)` and `des(1)` in the *SunOS Reference Manual*

7.2. Protecting Your Local Machine

This section describes security features that protect individual machines from unauthorized users. Many are standard UNIX security features, as introduced in the SunOS User Guide series. However, some are applicable only if the C2 security package is installed, as described in *Administering C2 Security*. If you administer a time-sharing system or server on a network requiring tight security, you will have to set up the following security measures for machines that are your responsibility. In a more open environment, you should encourage individual users to implement these security measures on their local machines.

The next section addresses the needs of the person administering a local machine. This person may be a system administrator in a secure environment or the individual user of the machine, in a more open environment. Some of the information applies only to system administrators in a secure environment. The text will indicate when this is the case.

Password Security and the `/etc/passwd` File

The user name and its corresponding password are the most critical security barrier in the SunOS environment. The `passwd` administrative database contains user names, encrypted passwords, and other critical information. On a local machine, this database takes the form of the `/etc/passwd` file. The `passwd` file contains information about every system and user account that can locally log in to the machine. It contains information about important system accounts, as well. The `passwd(5)` man page completely describes this file; this subsection

covers the common features that you will most likely use.

When you log in to a Sun machine, the `login` program consults the `passwd` database and verifies the user name and password you supplied. (Refer to the `login(1)` man page for a full description of this command.) If the user name is not in the `passwd` database or the password is not correct for the user name, `login` will deny you access to the machine. When you supply a user name that is in the `passwd` database and its correct password, `login` grants you access to the machine.

`/etc/passwd` is present on all types of Sun computers, from diskless clients to any type of server. In a C2 secure environment, the user name and sundry information is kept in `/etc/passwd`, while the password itself is in the `/etc/security/passwd.adjunct` file. This is because the encrypted passwords in `/etc/passwd` can be decrypted with effort by a technically-sophisticated person, but only `root` can read `/etc/security/passwd.adjunct`.

Here is a sample `passwd` file for a networked standalone machine.

```
root:uYU722Lg5xk/U:0:1:Operator:/:/bin/csh
nobody:*:65534:65534:/:
daemon:*:1:1:/:
sys:*:2:2:/:/bin/csh
bin:*:3:3:/:bin:
uucp:eXs0qzRjUOS8Y:4:8:/:usr/spool/uucppublic:
news:*:6:6:/:usr/spool/news:/bin/csh
sync::1:1:/:/bin/sync
sysdiag:*:0:1:System Diagnostic:/usr/diag/sysdiag:/usr/diag/sysdiag/sysdiag
stefania:1Zm62edD008es:3747:10:Stephanie:/home/dancer/stefania:/bin/csh
azhar:8Lf65seK9I3kt:6008:10:Kathy:/home/dancer/azhar:/bin/csh
+::0:0:::
```

Each entry has this syntax.

```
username:password:uid:gid:gcoss-field:home-dir:login-shell
```

The `passwd(5)` man page completely describes each of the following parameters.

<i>username</i>	User or system account login name, also referred to as the <i>user name</i> .
<i>password</i>	Account encrypted password.
<i>uid</i>	Account numerical user ID. In the previous example, the root account's user ID is 0, while stefania's is 3747.
<i>gid</i>	Numerical ID of the group to which the account belongs. Note that the stefania and azhar user accounts above both belong to group 10.

<i>gcos-field</i>	User's real name and other identifying information to be printed in the user's mail message heading. In the example, only the user accounts and the system account sysdiag have information in the <i>gcos</i> field.
<i>home-dir</i>	Full pathname of the account's home directory. For example, account azhar's home directory is <code>/home/dancer/azhar</code> .
<i>login-shell</i>	Shell the account accesses upon login. In the previous example, both user stefania and azhar access the C-shell (<code>/bin/csh</code>) when they log in.

In the sample password file above, the system accounts listed below are actually default user names.

```
root
nobody
daemon
bin
sync
```

The administrative files in `/etc` are owned by the root account. Depending on the circumstances, various SunOS programs also run with the user names `root`, `bin`, `nobody`, or `daemon` as their owners.

The `nobody` account is assigned under certain conditions to users that access your machine over the network. The NFS and RFS network services use the `nobody` account for various purposes, as explained in *The Sun Network File System Service* and the *The Remote File Sharing Service*.

Please note that in prior SunOS releases the user ID assigned to `nobody` was -2. In Release 4.1, that user ID has changed to an unsigned number, 65534, as required for POSIX conformance. This change will not affect your system's performance in any way.

The user names `uucp` and `news` in the sample above are used by `uucp`. The system diagnostics programs run with the `sysdiag` user name.

The entries below are the accounts of users allowed to log in to this machine.

```
stefania:1Zm62edD008es:3747:10:Stephanie:/home/dancer/stefania:/bin/csh
azhar:8Lf65seK9I3kt:6008:10:Kathy:/home/dancer/azhar:/bin/csh
```

The passwd File and NIS

If your network uses NIS, you can edit your machine's `passwd` file so that it additionally uses the NIS password facilities. You do this by placing the following characters as the last entry in `/etc/passwd`:

```
+:0:0:::
```

Thereafter, if someone logs in with a user name that is not in the machine's `passwd` file, the `login` program will look for the name in the NIS maps. The chapter *The Network Information Service* explains NIS maps and how to use the `+:0:0:::` escape characters.

Procedures for Setting Up a New User's Password

When you install a new release of the operating system, SunInstall creates an `/etc/passwd` file on each machine configured. However, the default `/etc/passwd` file only contains the system accounts, including root *without* an encrypted password. Therefore, the first task to perform on a newly installed system is to log in as root and give root a password—as explained in *The System Administrator's Role*.

Thereafter, you must modify the local password file—unless you saved a copy of the previous `/etc/passwd`. You also have to modify `/etc/passwd` whenever you add or delete a user for an existing machine or add a new machine to an existing server's network.

The following procedures show how to modify `/etc/passwd`. They assume that you, as system administrator, perform them on a user's machine, or on your machine. The procedures also assume that you know the root password for the machine. If you don't, have the user participate by typing in the root password during the procedure. In a less security-conscious environment, encourage the users to run these procedures themselves.

1. Obtain basic information from the user that you need for the `passwd` file entry. This includes preferred user name (must be unique within your network domain), full name as the user wants it displayed in mail headers, and preferred login shell.
2. Become superuser and access the `/etc` directory.

```
#cd /etc
```

Note that if the machine is a diskless client, you can update its `/etc/passwd` file by accessing `/export/root/client_name/etc`. In this instance, you do not need to know the client's root password.

3. Edit `/etc/passwd` using your preferred text editor.
4. Create an entry for the new user on a separate line.
5. Type the user's requested login name, for instance,

```
shami.ra:
```

The colon after the user name is the delimiter used in the `passwd` file to indicate the end of a field.

6. Leave the password field blank by typing another colon.

```
shami.ra::
```

7. Add a user ID for shamira, according to your site's policies. Do not use a user ID that already exists in your network domain. Especially do not use 0 or 1, the user IDs for the root and daemon system accounts.

If your site does not have a policy for assigning user IDs, you have to create one. As one suggestion, some companies use a person's employee number in the user ID field of `/etc/passwd`. Terminate the user ID with a colon.

8. Add a group ID number for the group you want the person to be in. The group called `staff` is provided by default. Its group ID is 10.
9. In the `gcos` field, type the user's name as he or she requested for the mail header. Follow it with a colon.
10. Type the full pathname of the person's home directory, which should be `/home/machine_name/user_name`. Follow it with the delimiting colon.
11. Type the user's preferred login shell, for example, `/bin/csh` or `/bin/sh`.

Here is an example of a complete entry for new user `shamira`:

```
shamira::235:10:Marsha:/home/dancer/shamira:/bin/csh
```

12. Close the `/etc/passwd` file, then create a home directory for the new user.

```
# cd /home/dancer
# mkdir shamira
# chown user_id# shamira
```

13. Exit from the root shell, then log out so that the log in prompt is displayed.
14. Have the user log in to the client by supplying the new user name, then pressing `RETURN` when requested for a password.
15. Make sure the user runs the `passwd` command immediately, to add password to the user name.

The `passwd` File and NIS

If your network uses NIS, you can edit your machine's `passwd` file so that it additionally uses the NIS password facilities. You do this by placing the following characters as the last entry in `/etc/passwd`:

```
+::0:0:::
```

Thereafter, if someone logs in with a user name that is not in the machine's `passwd` file, the `login` program will look for the name in the NIS maps. *The Network Information Service* explains NIS maps and how to use the `+::0:0:::` escape characters.

Setting Up User Groups on the Local Machine

Traditional UNIX user groups consist of individuals who use the same set of files and are granted the same set of permissions. Typically these are individuals on the same project or in the same department. In the SunOS environment, you can implement two types of groups: user groups on the local machine and netgroups. *The SunOS Network Environment* explains how to set up netgroups. This next subsection explains user groups on the local machine.

Local user groups are most effective for security-conscious sites using time-sharing systems with terminals attached. You may also want to set up local user groups for a security-conscious environment where each machine has more than

one user locally logging in to it. If you administer this type of machine, consider implementing groups if a subset of these users must access the same set of sensitive files. For example, you might implement groups on an accounting department's machine, where some users can access certain files that others are denied.

In these environments, you as system administrator must establish these groups, as opposed to an individual who happens to be the most frequent user of the machine.

The /etc/group File

In the SunOS environment, the basic form of group protection is the `group` database. On your local machine, this database takes the form of the `/etc/group` file. (Refer to the `group(5)` man page for detailed information about this file.) Each entry in `/etc/group` has the following syntax:

```
groupname:password:gid:user-list
```

The fields are defined as follows.

<i>groupname</i>	Name of the group.
<i>password</i>	Encrypted group password, if any.
<i>gid</i>	The group's numerical ID. It must be unique on the local machine.
<i>user-list</i>	List of users in the group.

As in the `passwd` file, a colon terminates each field.

When you install a new release, SunInstall creates the following group file:

```
wheel:*:0:
nogroup:*:65534:
daemon:*:1:
kmem:*:2:
bin:*:3:
tty:*:4:
operator:*:5:
news:*:6:
uucp:*:8:
audit:*:9:
staff:*:10:
other:*:20:
+:
```

SunOS system accounts and executing programs belong to most of the groups shown above. The root account belongs to the `wheel` group, which has the `gid` 0. Members of the `operator` group have special permissions that enable them to run certain commands without being superuser. These commands are `dump`, `restore`, and `shutdown`. Members of the `operator` group can run these commands on remote machines without being in the remote machines' `.rhosts` files. If your site employs certain individuals whose job it is to specifically run daily dumps and restores, you might want to add these people to the `operator`

group.

As with the `nobody` account in the `passwd` file, the `nogroup` group is assigned under certain conditions to users that access your machine over the network. In prior SunOS releases, the group ID assigned to `nogroup` was `-2`. In Release 4.1, that group ID has changed to an unsigned number, `65534`, as required for POSIX conformance.

The plus (+) entry at the end of the example indicates that if a user's group is not in the local `group` file, the machine should check the NIS group database. Refer to *The Network Information Service* for more information.

Determining Group Membership

You can find out which groups you are in by using the `groups` command. To display group names for your own account, simply type the following.

```
% groups
staff
```

To display group names for another user on your local machine, type the following:

```
% groups user_name
user_name : staff
```

Creating New User Groups

On a newly-installed system, SunInstall provides two default groups: `staff` and `other`. As superuser you can create additional groups and assign them members by entering the information in the `/etc/group` file. Since group membership is determined at login time, if you add yourself (or are added) to another group, you need to log out and log in again before you actually join that group.

Below are procedures for adding local user groups to your machine:

1. Become superuser on the machine.
2. Edit `/etc/group` using your preferred text editor.
3. Create an entry for the new group.

```
duet:*:55:stefania,azhar
```

Here `duet` is the name of the new group. The asterisk in the next field indicates that `duet` does not have a password. Passwords are not recommended for groups, although you can set them up, if required. Be aware that no equivalent to the `passwd` command exists for changing the group password.

The `55` in the third field is the group ID; it must be unique on the local machine. If your network runs NIS, the group ID must be unique throughout the domain.

The final field includes the login names of the members of group `duet`. The example above includes users `stefania` and `azhar`, both local users of a

machine called “raks.” It does not include user shamira, who also uses raks.

4. Close the group file.
5. Have the members of the new group log out and log in again to have the additional group membership take affect.

The next time the members of the new group log in to the machine, they will be members of their new group as well as their default group.

Changing a User’s Group ID

When a user creates a file, the operating system assigns a group ID to the file, which is the *effective group ID* for the user’s account – if the directory in which the file is created does not have its `setgid` bit set. For more information about directories’ `setgid`, see the later subsection *How setgid Affects Directories*. A user can belong to a number of groups, but all files that he or she creates are assigned the effective group ID. Initially, a user’s effective group ID is the one listed for his or her account in the `/etc/passwd` file. Thus all files that users create have the effective group ID 10—group ID of the default group staff—unless it is changed.

If you create additional groups on the local machine, users might want to create files owned by one of the new groups, rather than by staff. They can do this by using the `newgrp` command to change their effective group IDs to the IDs of the preferred groups.

`newgrp` enables the user to obtain a new effective group ID at any time. Its syntax is as follows.

```
newgrp [ - ] [ group_name ]
```

The `newgrp(1)` man page contains a complete technical description of the command.

Individual users should run `newgrp` for their own accounts. Note that a user must already be listed in the group database as a member of the group he or she specifies to `newgrp`. After the user runs `newgrp`, it starts up a new shell with the new effective group ID in effect. Note that the group ID listed in the `/etc/passwd` file is not changed when you run `newgrp`.

The dash option of `newgrp` means change the effective group ID to that specified in the `/etc/passwd` file.

7.3. Protecting Files on the Local Machine

You protect files on a local machine by implementing *discretionary access control*. This security feature provides data security by enabling users, at their discretion, to give away access to files and directories. It is a traditional UNIX feature implemented in SunOS Release 4.1 and introduced in the SunOS User Guide series. Discretionary access control involves assigning file access permissions. Only the owner of the file or the superuser can assign and modify file permissions.

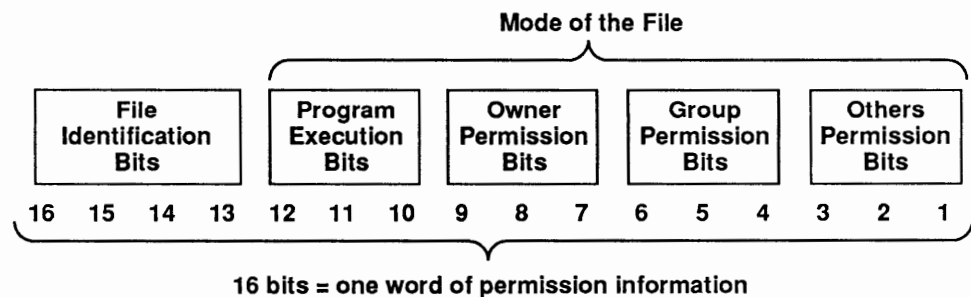
File permissions govern who can read, write, or execute a file. One file might contain sensitive information that only the owner should have permission to read. Another file might contain public information that everyone can read, but only the owner has the right to modify. Yet another file might have project-wide information that members of a group should all be able to modify. All these files may co-exist simultaneously because users have discretionary access control on a file-by-file (and on a directory-by-directory) basis.

This section assumes you are familiar with the basic UNIX file permissions—read (r), write (w), and execute(x)—and how they apply to files and directories. It also assumes that you understand the three categories of users to whom you assign file permissions: owner, group, and others. If you are unfamiliar with these concepts, refer to *SunOS User's Guide: Getting Started* for further instruction before continuing with this chapter.

How File Permissions Are Stored

The *inode* (information node) of a file or directory contains its file permissions, along with other vital information. This information is stored within the inode as a 16-bit word. The first nine bits of the word contain data representing the actual file permissions (r,w,x) or indicate that file access is denied (-). The next three bits give information relevant to file operation if the file is an executing program. Together, these 12 bits comprise the *mode* of the file. The remaining four bits specify file type and are set when the file is created. Figure 7-3 illustrates how permission bits are grouped within the 16-bit word.

Figure 7-3 *Format of Permission Bits*



Note how each category of user (and executing programs) is assigned three permission bits within the 16-bit word:

Others	uses bits 1-3
Group	uses bits 4-6
Owner	uses bits 7-9
Program Execution	uses bits 10-12

These three bit combinations are called *triplets*.

Changing File Permissions with `chmod`

You use the `chmod` command to change permissions on a file or directory. This is the basic syntax of `chmod`.

```
/usr/bin/chmod [ -fR ] mode filename
```

The `chmod(1v)` man page completely describes the `-f` (force) and `-R` (recursive) options. The *mode* parameter represents a set of file permissions, expressed in either symbolic or absolute mode. *filename* represents the name of the file or directory to which these permissions apply.

As shown in the syntax statement, `chmod` sets permissions in either of two modes: symbolic or absolute. *Symbolic mode* uses combinations of letters and symbols. *SunOS User's Guide: Getting Started* introduces symbolic mode. The next section explains how to use *absolute mode*, which uses octal numbers to represent file permissions.

Note: In absolute mode, you also specify octal values for the program execution triplet, but they have a different meaning, as shown shortly.

When you use `chmod` in absolute mode, you specify numeric values for the triplets representing owner, group, and others, thusly.

```
/usr/bin/chmod nnn filename
```

where *nnn* is a 3-digit number with the format:

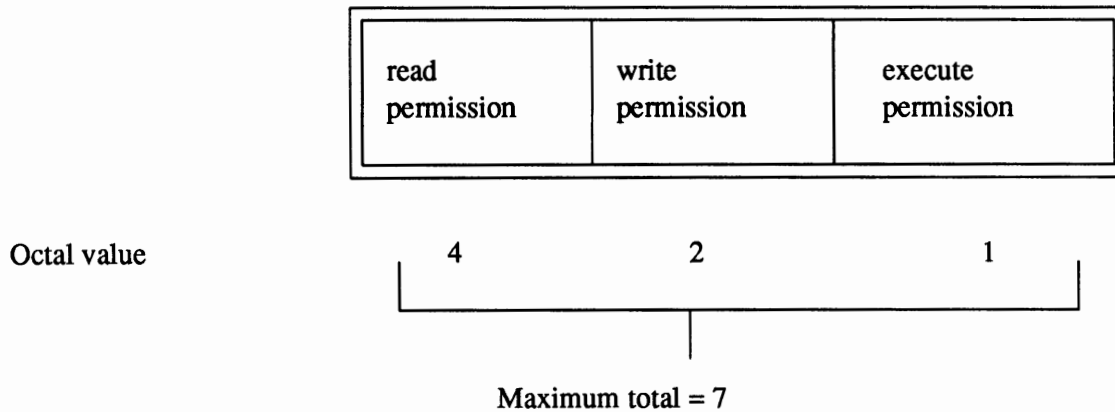
<i>n</i>	<i>n</i>	<i>n</i>
owner	group	others
triplet's	triplet's	triplet's
value	value	value

For example, the following command: uses the absolute mode `666` to assign read-write permission to file `zills` for all categories of user.

```
# chmod 666 zills
```

The permission assigned by `chmod` to each triplet turns on (or does not turn on) one or more of its three permission bits. Figure 7-4 depicts the format of permission bits for a triplet and the values assigned to them.

Figure 7-4 *Octal Values for the Bits in a Triplet*

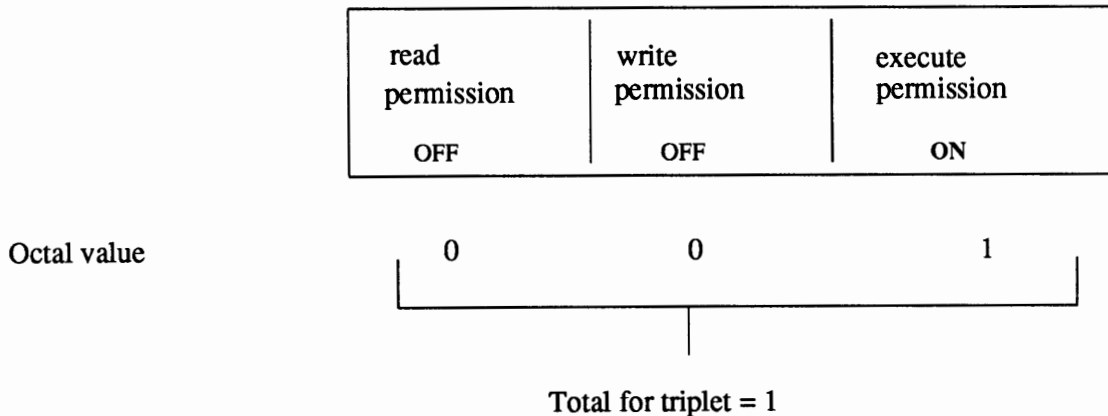


If you specify a 1 to `chmod` for a particular user category, this turns on only the furthest right bit of the group and others triplets.

`chmod 711 filename`

Then members of the group owning the file and the public have execute-only permission for the file. The following illustration shows how bits in the others triplet are turned on or off for execute-only (“1”) permission.

Permissions for “Others”



However, if you specify a 5 for these categories, this turns on the furthest left bit and the furthest right bit in the group and others triplets.

`chmod 755 filename`

Now group members and the public have read and execute permission for the file.

The maximum value you can specify for a triplet is 7. In the examples above, `chmod` specifies this value for the owner triplet. This turns on all three bits in the triplet, for a total octal value of 7, and grants the owner read, write, and

execute permission.

The minimum value you can specify to `chmod` is zero. This turns off all three bits of the triplet and denies access to users in the corresponding category.

Here is a list of all octal values that you can set with `chmod`.

For the Owner Triplet:

```
400    Read by owner.
200    Write by owner.
100    Execute (search in directory) by owner.
```

For the Group Triplet:

```
040    Read by group.
020    Write by group.
010    Execute (search) by group.
```

For the Others Triplet:

```
004    Read by others.
002    Write by others.
001    Execute (search) by others.
```

For the Program Execution Triplet:

```
4000   Set user ID on execution.
        (ignored if the file is a directory)
2000   Set group ID on execution
1000   Set the Sticky bit
```

For the program execution triplet above, the values in the two leftmost boxes, `setuid` and `setgid`, are explained in the later section, *Real and Effective User and Group IDs*. The subsection *Setting the Sticky Bit* explains the sticky bit value.

Absolute Mode Made Easy

Due to its use of octal values, absolute mode seems harder to use at first than symbolic mode. But actually, it's easier in a lot of ways. The trick is to memorize the permission values and permissions that they enable. Tables 7-1 and 7-2 show commonly used absolute modes.

Table 7-1 *Common Permission Modes for Files*

<i>Mode</i>	<i>Permissions Granted</i>
640	Owner & group can read; only owner can write
600	Owner can read & write; group & others denied access
755	Everyone can read & execute; only owner can write
750	Owner & group can read & execute; only owner can write
711	Owner can read, write, & execute; everyone else can only execute
700	Only owner can read, write & execute files

Table 7-2 *Common Permission Modes for Directories*

<i>Mode</i>	<i>Permissions Granted</i>
640	Owner & group can read; only owner can write
600	Owner can read & write; group & others denied access
755	Everyone can enter & list; only owner can create files
750	Owner & group can enter & list; only owner can create files
700	Only owner can enter, list, and create files

Here are some examples showing common uses of `chmod` in absolute mode.

Sharing a file with members of your group

To do this, you need to make your home directory accessible and the file readable with the following commands:

```
% chmod 750 $HOME
% chmod 640 filename
```

where the `$HOME` variable pulls in the path for your home directory.

Allowing everyone on the machine to access your file

To do this, you need to issue these commands:

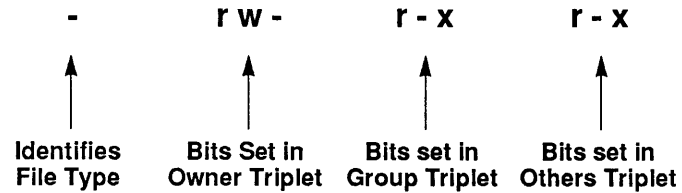
```
% chmod 755 $HOME
% chmod 644 filename
```

Displaying File Permissions with `ls -l`

If you or others on the local machine are denied access to a file or directory, you need to check the permissions assigned to it and determine who owns it. The `-l` option of the `ls` command lists information about a file or directory, including its owner and current file permissions. (The `ls(1v)` man page contains full technical information about this command.) When you type `ls -l`, the resulting display resembles the following for the file called “whispers.”

```
% ls -l whispers
-rw-r-xr-x 1 stefania      30 Dec 27 17:12 whispers
```

The *SunOS User's Guide: Getting Started* explains this display in detail. The next figure shows how the permissions displayed by `ls -l` are organized and how they relate to the permission bits set by `chmod`.



The dash in the first column identifies the file as an ordinary file. The owner has read-write permission for the file, but cannot execute it. The members of the group owning the file and the public (others) have read and execute permission, but cannot write to the file.

You can determine the absolute mode of the file by adding up the octal values for each triplet. For example, the owner triplet has the value 600.

```
r = 400
w = 200
- = 0
```

The octal value for the group triplet is 050.

```
r = 040
- = 000
x = 010
```

The others triplet has the value 5.

```
r = 004
- = 000
x = 001
```

Thus the absolute mode for file `whispers` is 755.

Changing Ownership of a File

To change ownership of a file, you use the `chown` command.

```
/usr/etc/chown [ -fHR ] owner[.group] filename ...
```

The `chown(8)` man page fully describes the parameters above. Only the superuser can use `chown`.

The most common form of this command is used for changing ownership of a file.

```
# chown tut filename
```

Here the user `tut` is designated as the file's new owner. Of course, the `passwd` database must have an entry for `tut` in order for `chown` to work. You can also supply a numeric user ID for the *owner* parameter or numeric group ID for the *group*.

Changing Group Ownership of a File

The owner of a file and the superuser can change the group that owns a particular file by using the `chgrp` command. For example, suppose you wanted to change ownership of the file `whispers` from the group `staff` to the group `mktg`. You would use the following command.

```
% chgrp mktg whispers
```

In order to do this, the file's owner must be a member of the group `mktg`, and the group `mktg` must be defined in the `group` database.

After you change the file's ownership with `chgrp`, the group ID of the file changes from the previous group ID to the group ID assigned to `mktg`.

Assigning Permissions at File Creation with `umask`

All UNIX-based systems give each newly-created file a default set of permissions. This set of permissions is defined in a *file creation mask*. SunOS Release 4.1 uses a default file creation mask of 22, which is set by the `init` daemon when your machine boots. With this file creation mask set, each file grants full permissions to the owner and read and execute permissions to the group and others. If individual users require tighter file security, they can change the file creation mask by using the `umask` command.

The *SunOS User's Guide: Getting Started* explains how to use `umask`. Additionally, the `csh-builtins(1)` man page gives full technical information about `umask`, in addition to other useful built-in commands.

If your site requires very tight security, encourage users to set a default file creation mask of 77, denying access rights to anyone but the file's owner. The new file creation mask only affects the mode of newly created files. It does not prevent file owners or you as superuser from changing permission modes with `chmod`. Nor does it prevent users from changing the `umask` values in their `.cshrc` or `.profile` files.

Encrypting Files

Placing a sensitive file in an inaccessible directory (700 mode) and making the file unreadable by others (600 mode) will keep it secure in most cases. However, if someone guesses your password or the `root` password, they can read and write to that file. Also, note that as system administrator, you already have permission to read and write to any file on the system, even though to do so would be considered unethical. Also, the sensitive file gets preserved on backup tapes every time you do a full dump.

Fortunately, you have an additional layer of security available to all SunOS users: the optional file encryption kit available as a software package from SunInstall. The file encryption kit includes the `crypt` command. This

command uses rotor encryption, which means a key is rolled through the text, disguising its characters. The advantage of the rotor encryption algorithm is that the text can be decrypted the same way by the same key. Unfortunately, rotor encryption is vulnerable to attack simply by examining encrypted text for patterns to discern the key. Once a would-be invader determines the key, the text can easily be decrypted. The longer your key, the more difficult it is for invaders to decipher. However, the operating system truncates keys longer than eight characters.

The *SunOS User's Guide: Getting Started* explains how to use the `crypt` command. For a detailed technical description, refer to the `crypt(1)` man page.

DES Encryption

For extremely sensitive files, you can use the `des` command for greatly improved data security. This technique works best for relatively stable material. The disadvantage of DES encryption is that you can't use the `-x` option of an editor to modify the data. On the other hand, a DES-encrypted file is almost totally secure, unless someone can guess the key. Unlike `crypt`, which uses the same algorithm both ways, `des` has different flags and algorithms for encrypting and decrypting.

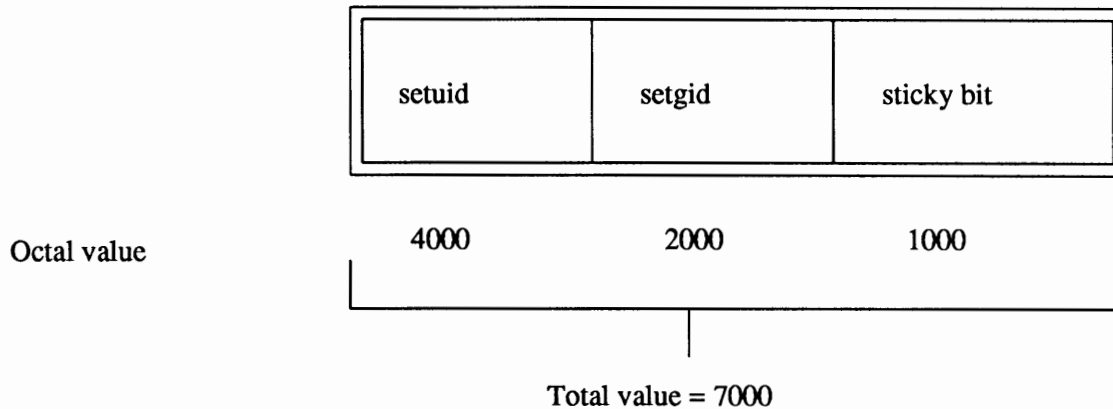
The *SunOS User's Guide: Doing More* contains instructions for using the `des` command. The chapters *The Sun Network File System Service* and *Administering C2 Security in System and Network Administration* also discuss DES encryption. For complete technical details, refer to the `des(1)` and `des(4)` man pages.

7.4. Setting Security Measures for Executing Programs

The `chmod` command not only lets you define file permissions but also lets you assign octal values that affect how running programs operate. These values turn on the bits of the program execution triplet of the 16-bit file information word of the file's inode, as shown earlier in Figure 7-3. The values set the following.

- The sticky bit
- The effective user ID bit (`setuid`)
- The effective group ID bit (`setgid`)

Figure 7-5 illustrates the format and values of these bits.

Figure 7-5 *Format of the Program Execution Triplet*

Setting the sticky bit primarily affects the performance of programs and is not a security measurement, per se. On the other hand, setting the effective user and group IDs are important security features, though they are meaningful only for executable files. Setting the group ID is also meaningful for directories.

Setting the Sticky Bit

In SunOS Release 4.1, you can mark a file or directory for special treatment by setting its sticky bit. You cannot delete a file with the sticky bit set unless you are its owner, the superuser, or have write permission for it. Moreover, you cannot delete files from a directory with the sticky bit set unless you are its owner, have write permission for the directory, or are the superuser.

The use of the sticky bit in Release 4.1 differs from the traditional UNIX usage. Traditionally, setting the sticky bit indicated that the file's process image should remain in swap space, even when it isn't running. This is not the case for SunOS 4.1. The `sticky(8)` man page explains the SunOS implementation of this feature.

As shown in Figure 7-4, the sticky bit is the right-most bit of the program execution triplet and has the octal value of 1000. Only the program's owner or you as superuser can set this bit. Type the following.

```
# chmod 1nnn program_name
```

where 1 is the value that turns on the sticky bit for `program_name` and `nnn` is the program's file permissions. The entry below sets the sticky bit for file `singing_program`.

```
# chmod 1755 singing_program
```

You can also use the symbolic mode command to set the sticky bit for the file.

```
# chmod +t singing_program
```


For directories, the sticky bit indicates that only a file's owner and the superuser can remove it. (This is useful for public directories such as `/tmp`).

Real and Effective User and Group IDs

When you execute a program, the SunOS kernel creates a new process and assigns that process two pairs of ID numbers. They are the real and effective user ID, and the real and effective group ID. The operating system uses the *effective* user and group IDs to determine access permissions for the process. The *real* user and group IDs are used for accounting purposes. The real and effective IDs of a process are usually the same, unless the executing program has the `setuid` and/or `setgid` bits set. An executing program with the `setuid` bit on is typically referred to as a `setuid` process, or as “running `setuid`” to a particular user.

If the effective user ID of the process is the same as that of the executable file's owner, that process has the owner's access permissions. If the effective group ID of a process matches that of the file's group, or if the process' grouplist contains the file's group, that process has the group's access permissions. Otherwise, that process has the access permissions of others.

When you execute a regular program, the kernel creates a new process to carry out the tasks of that program. The system assigns your user ID and group ID to the new process as its real and effective user and group IDs. (The executable file from which the process was created retains its current set of permissions.) The executing process has the same access to a file that you do. For example, if the process needs to write to a file, you must have write permission for that file. If a process needs to create a file in a directory, you must have write and execute permission for that directory.

By contrast, when you execute a program with the `setuid` bit turned on, the resulting process and its children have the effective user ID of the program's owner, not your user ID. The real user ID of the `setuid` process is the same as your user ID. Likewise, when you execute a `setgid` program, the resulting process and its children have the effective group ID of the program's group. Consequently, these processes have the same access permissions as the owner of the program would have, no matter who executes that program.

Whether to turn on the `setuid` and `setgid` bits is an important security consideration. Many programs come originally with either or both bits set, as described in the next subsections. As system administrator, you should give careful consideration to this condition and perhaps modify it. In addition, people creating programs may ask you to enable this option for their own applications. Once again, careful consideration is in order.

Setting the User ID

Any user can turn on the `setuid` bit for an executable program, IF he or she owns the program. To do this, you use the following form of `chmod`.

```
chmod 4nnn program_name
```

The value 4 turns on just the `setuid` bit, the leftmost bit of the executing program triplet, as shown in Figure 7-4. *nnn* represents the file permissions assigned

to *program_name*. To turn on the sticky bit for the program and make it run `setuid`, specify the octal value `5nnn` (4000 bit + 1000 bit).

If you prefer, you can use the symbolic mode command to accomplish this.

```
chmod o+s program_name
```

Then, if necessary, use the `chown` command to specify the login name that will own the program. Refer to `chmod(1)` and `chown(1)` in the man pages, if you need help with these commands.

As an example, suppose you have data that only you have permission to read, but you want to provide a way for others to look at selected parts of the data. You could write a `setuid` program to allow this, then turn on the program's `setuid` bit as follows.

```
% chmod 4711 program
```

The permissions `4711` enable all users to execute your program. Because the resulting process is `setuid`, it runs with your effective user ID, not the ID of the person who executed it. Therefore, the process has your permissions for whatever files and directories it touches.

To turn off `setuid` permission for the example above, use `chmod` and change the octal value to `711`, `751`, or `755`, without the leading `4`. Writing `setuid` programs is covered in *System Services Overview*.

Note: In general, `setuid` programs are a security problem, especially when they set the user ID to `root`.

When you copy a `setuid` program owned by somebody else, the `setuid` bit remains set, although the ownership changes. When you change the group of a program with `chgrp`, however, the `setuid` bit gets turned off. The exception to this is `root`, who can change group or owner without altering the `setuid` bit.

Setting the Group ID

The set group ID mechanism (`setgid`) is similar to the `setuid` mechanism, but works for groups rather than individual users. In general, `setgid` programs are more secure than `setuid` programs, because group permissions are usually a subset of user permissions.

You can make a program `setgid` (to one of your group IDs) if you are a member of that group. For example, if you have data that cannot be read by others outside of your group, you can write a `setgid` program to allow this, as shown in the *System Services Overview*. Then, turn on the program's `setgid` bit as follows:

```
% chmod g+s program
```

This turns on the `setgid` bit, which is the center bit of the executing program triplet.

System Programs That Are setgid

You can list some system programs that are setgid by issuing this set of commands.

```
% ls -lag /usr/bin | grep r-s
8 -rwxr-sr-x 1 root operator 7536 Oct 15 16:08 df
11 -rwxr-sr-x 1 root kmem 10424 Oct 15 16:08 iostat
16 -rwxr-sr-x 1 root kmem 16384 Oct 15 16:08 ipcs
33 -r-sr-xr-x 1 root bin 33047 Oct 15 15:56 nsquery
1 drwxr-sr-x 2 bin staff 1024 Nov 2 17:05 sunview1/
6 -rwxr-sr-x 1 root tty 5168 Oct 15 15:52 wall
16 -rwxr-sr-x 1 root tty 16384 Oct 15 15:52 write
```

Two of these programs are setgid kmem because they need to read /vmunix and /dev/kmem, which, for security reasons, cannot be read by the general public.

The df program is setgid operator, so that it can read disk partitions to see how much space they have left. The wall and write programs don't need to be setgid tty unless terminal devices are group tty, which they are not by default.

A program may have both the setuid and setgid bits set. The printer commands are examples, as you can see by issuing the following set of commands:

```
% ls -lag /usr/ucb/lp* | grep rws
24 -rws--s--x 1 root daemon 24576 Oct 15 12:40 lpq
24 -rws--s--x 1 root daemon 24576 Oct 15 12:40 lpr
24 -rws--s--x 1 root daemon 24576 Oct 15 12:40 lprm
```

All the programs need to be setgid to daemon, because the daemon group should own the /var/spool directories for each output device. The lpr program is setuid to root, so that it can access all the necessary files in the print spool area of /var/spool. Actually lpq and lprm do not need to be setuid root.

How setgid Affects Directories

For the sake of System V compatibility, the notion of setgid is extended so that it applies to directories also. UNIX System V permits users to belong to only one group at a time. BSD 4.2, on the other hand, permits users to belong to multiple groups. Prior to Release 4.0, the SunOS operating system followed BSD in this respect. As a consequence, files took the group of their parent directory since their creator might be a member of more than one group.

With SunOS Release 4.1, files created on file systems not mounted with the grpuid option obey System V semantics. (See the mount(8) man page for a description of the -grpuid option.) That is, their group ID is set to the effective group ID of the creating process. You can alter this behavior inside any directory by turning on the setgid bit of that directory. By default, file systems are mounted with System V semantics, although on the standard release all directories have the setgid bit enabled, thus preserving 4.2 BSD group behavior. Files created on file systems mounted with the grpuid option obey 4.2 BSD semantics. That is, they inherit the group ID of their parent directory. It is impossible to alter this behavior on a per-directory basis.

To clear the `setgid` bit from a directory, use the `g-s` option of `chmod` as follows:

```
% cd
% ls -ldg
drwxr-s--- 27 tut   staff      1536 Aug 27 14:12 .
% chmod g-s .
% ls -ldg
drwxr-x--- 27 tut   staff      1536 Aug 27 14:12 .
```

Note that absolute mode values to `chmod`, such as `0750` or `2750`, do not work to clear or set the `setgid` bit on directories.

To summarize, a file's group ID is set to the effective group ID of the process, if the file system was not mounted with the `grpuid` option of `mount`, and the `setgid` bit of the parent directory is clear. Otherwise, a file's group ID is set to that of the directory in which the file is created. Take the group behavior at your installation into account when deciding which groups to set up.

How SunOS Commands Affect User and Group IDs

Certain SunOS file handling commands affect whether `setuid` and `setgid` permissions are preserved. For example, when you copy a file with `cp`, the permissions of the source file are duplicated if the destination file doesn't already exist. However, note that if a destination file already exists, its original permissions are retained. `setuid` is preserved, as is `setgid` permission, but ownership changes to the user making the copy. To demonstrate this, try the following.

```
% cp /usr/bin/su .
% ls -lsg su
7 -rwsr-xr-x 1 stefania staff 6280 Jan 4 17:33 su*
```

When you move a file with `mv`, the file's permissions, including `setuid` and `setgid` bits, aren't changed at all, but ownership does change. To demonstrate this, try the following.

```
% chmod 4755 su
% mv su mysu
% ls -lsg mysu
7 -rwsr-xr-x 1 stefania staff 6280 Jan 4 17:33 mysu*
```

When you make a hard link to a file with `ln`, the new linked file has the same `setuid` and `setgid` permissions as the original file. To demonstrate this, try the following:

```
% ln mysu lnksu
% ls -lsg lnksu
7 -rwsr-xr-x 2 stefania staff 6280 Jan 4 17:33 lnksu*
```

Note the link count 2 in the third field. At this point, removing `mysu` does not get rid of `lnksu`, which continues to have `setuid` permission. The moral: be sure to check the link count of `setuid` or `setgid` programs before removing them. If the link count is greater than one, change the file's mode to `000`, and

then remove it. Changing the mode of the original file changes the modes of all linked files as well, rendering them harmless even if they continue to exist.

The next section, *The Administrator's Guide to Safe Systems*, gives tips for protecting `setuid` and `setgid` programs.

7.5. The Administrator's Guide to Safe Systems

As SunOS system administrator, you may take care of a system configuration ranging from a single machine to a network of disparate computers. Your routine includes starting them up, shutting them down, adding new users, performing backups—the full range of activities covered in this text. Underlying all these tasks is your responsibility to ensure the integrity and privacy of data.

This section explains how to use the commands and files presented earlier in this chapter to prevent security breaks, including these subjects.

- An overview of your system security responsibilities.
- Types of computer “maladies” that unauthorized users may impose upon your system or network.
- Tips and tricks for making individual user accounts secure.
- Suggestions for making the file systems secure.
- Suggestions for making a site physically secure.
- Suggestions for handling security encroachments once they have happened.

Security Administration

As system and/or network administrator, you should always keep the following important security goals in mind.

- Preventing unauthorized access. People who are not authorized to use a system should be kept off it. You can ensure this by educating users as to good password selection, managing passwords intelligently, setting up login reporting, and selectively auditing user activities.
- Maintaining system integrity. Computer systems should be fast, accurate, and reliable. You can ensure this by backing up file hierarchies on a regular basis, running `fsck` after system crashes, fully testing software before installing it, and upgrading hardware whenever it starts to manifest problems.
- Preserving data privacy. Users should be able to keep sensitive data private. You can ensure this by making sure users set up file permissions and data encryption, `su` reporting, and periodic file system audits.
- Preventing interruption of service. Computer systems should not be impaired by users who deliberately try to use up resources. You can ensure this by setting up appropriate disk quotas, periodic monitoring of network and CPU activity, and consideration of others.

This list is given in order of importance. If the first goal is not met, the others will never follow.

Password Administration

Password administration involves setting up users' passwords and making sure they are changed. This subsection gives tips for effective password administration.

In a security-minded environment, password administration should always be the responsibility of one person: you, as system administrator. If your site fits this description, then you must assume the responsibility for setting up passwords and, perhaps, the entire user environment, for everyone. Therefore, you should pay particular attention to the tips below and set them up as your site requires.

In a more open environment, each user typically sets up the local password and operating environment for his or her machine. As system administrator, your responsibilities in this area should be in an advisory capacity. If NIS runs on your network, you may be responsible for `yppasswd` administration, as explained in *The Network Information Service*. Therefore, you should read the tips below and implement those that are appropriate.

- Set up the local `/etc/passwd` files for all machines at your site.
- Run the `passwd` command for each user, having them type in their passwords.
- Set up password aging, if needed at your site. See the section, *Keeping System Directories and Files Secure*, for more information.
- Check the local `passwd` files on a regular basis, and make sure that all accounts have passwords. This includes all system accounts except for `sync`, which does not need one.
- Change the password on "guest" accounts regularly. Some sites set up guest accounts for visitors from other companies or departments who use a machine for a brief period of time. If this pertains to your site, you should make sure that the password on the guest account changes after the guest user leaves.
- Make sure that when a person no longer works at your site, his or her password immediately gets removed from the appropriate `passwd` databases. This should include not only the local `/etc/passwd` file on any machine the person used but the NIS maps, as well.
- Set up the `umask` and `PATH` environment variables for each user—appropriate only for very secure environments.

Safeguarding the Superuser Privileges

Only an individual logged in as superuser can run most administrative commands. The superuser account has a user ID of zero, can read and write *any* file, or run *any* program, regardless of permissions on these files or programs. Normally, the user name `root` has superuser privileges, since its user ID is zero. But the operating system also regards any other user name with a user ID of zero as a superuser. Since the superuser has permission to read or write any file, and to execute any program, the possibilities to mess up are much greater than for ordinary users.

Secure systems should have only one login name— `root` — with a user ID of zero. On nonsecure systems it is possible to log in as `root`, but you should discourage this practice. For example, if your Sun computer configuration has several system administrators, it is impossible to tell which one logged in as `root`. However, when you use the `su` command to become superuser, your user name is recorded on the console and in the audit trail. Moreover, if system crackers guess the superuser password, they can log in without leaving any evidence that they have done so.

For these reasons, you may want to consider taking out all the `secure` keywords in your `/etc/ttytab` file. With this file thus modified, your machine will reject all `root` logins, but you can still gain superuser access with the `su` command. (Refer to the *Adding Hardware to Your System* chapter and the `ttytab(5)` man page for more information about secure logins.)

Also, if the `/dev/console` entry in `/etc/ttytab` does not have the `secure` keyword, you will be required to enter the `root` password whenever you boot the machine in single-user mode. Thus, deleting the word `secure` from the `/dev/console` entry in machines that are not physically secure (not located where you can be sure that only trusted personnel can get at them) is a further security measure. It prevents someone from cracking into the machine by entering the `L1-a` aborting sequence, then booting single-user.

Here are some steps you can take to protect the superuser account.

- Don't run other users' programs as `root`; switch users from your account to theirs with `su` instead.
- Don't ever put the current directory first in your `PATH`, as elaborated on shortly.
- Get in the habit of typing `/bin/su` instead of merely `su`. The full path-name will guarantee that you don't run a fake `su` Trojan horse, as discussed shortly. Even better, get in the habit of typing `/bin/su -`, so that the superuser's environment is only `root`'s.
- Don't leave your workstation or terminal unattended, especially when you have a `root` shell running in the console or other window. Always invoke `lockscreen` when you go away from your workstation.
- Change the `root` password often, and be very good about password selection.
- If you are doing security auditing, audit every invocation of `su`, and inspect these audit records periodically.
- Don't let anyone run as superuser, even for a few minutes, even if you're watching.

A Rogue's Gallery of Computer Invaders

Computers succumbing to unwanted invasion has become an all-too-familiar, even news-worthy event in recent years. This subsection describes some of the more common types of computer invaders and diseases. Later parts of the chapter present measures you can take to protect your system from this type of encroachment.

Invasion by Trojan Horses

A Trojan horse is a program that appears to perform a useful function while doing something nasty behind the user's back—such as compromising system security or destroying disks. Computer invading *Trojan horses* get their names from the Trojan War myth, where the Greeks pretended to abandon the siege of Troy, leaving behind a large wooden horse. The Trojans, regarding the horse as a sacrifice to the gods, opened the city gates and took it into Troy. Later that night, Greek soldiers concealed in the horse opened the gates to the Greek army, who conquered the city. In *The Irish Genius*, Woody Allen summarizes the incident:

Two thousand years have passed
since bold Priam said,
“Don't open the gates,
who the hell needs a wooden horse
that size?”

– Shawn O'Shaun

One example of a Trojan horse might be an unscrupulous system administrator rewriting the system's `crypt()` library routine. The corrupted `crypt` would mail him or her the login name, current directory, and typed-in key every time the library routine gets called. Thus, the system administrator would not only find out everybody's password, but would also know the keys (with locations) for all encrypted files on the system.

This manual assumes that you are a trustworthy system administrator who would never consider such a thing. But suppose you administer a system attached to a large network. You have no assurance that other administrators on that network can be trusted. You also have to trust your UNIX system vendor, because unscrupulous vendors could place Trojan horses on every system they sell.

Regular users who aren't entrusted with the administrator's responsibilities can also place Trojan horses. For example, writing a substitute `su` that could be placed in a public directory where you, as system administrator, or anyone else might run it, is fairly simple. Such a script would look much like the regular `su`. Since it removes itself after execution, it's hard to tell that you've actually run a Trojan horse, rather than just accidentally mistyped your password.

Setting your search path so that the current directory comes last will prevent you from running this kind of Trojan horse. Also, get in the habit of running `su` from your home directory. Unfortunately, even if you have a secure search path, it's still possible to run Trojan horses that aren't named after system utilities. For `root`, the search path probably shouldn't contain the current directory at all.

Trojan Mules

A Trojan mule is a kind of Trojan horse, but is executed by somebody else and left around as a trap. For example, someone could write a program that imitates `login`, and run it on an unattended terminal. When somebody tries to log in on that terminal, the program would mail the user name and password to the program's author, print a message saying `login incorrect`, and exit. At this point, the victimized user would see the real `login` program, and think that problem was caused by a simple typo.

Trojan mules differ from Trojan horses in that they don't pretend to do anything useful. Also, Trojan mules usually run once, then remove themselves from the system, whereas Trojan horses can be run again. This makes Trojan mules less of a threat than Trojan horses.

Computer Viruses

A computer virus is the worst kind of Trojan horse. In fact, some of the more infamous viruses have received nation-wide coverage by the United States news media. A computer virus infects a system by attaching itself to other programs and converting them into viruses.

Viruses are classified as "malignant" or "benign." Benign viruses replicate themselves, but do not intentionally cause damage. They generally cause only mischievous behavior, such as ringing the workstation's bell, or displaying an unexpected cartoon when the user locks the machine's screen. But malignant viruses not only replicate themselves but also try to cause damage.

Computer viruses can spread quickly, particularly if you run an infected program as `root`. A recent experiment demonstrated that a virus could usually gain `root` privileges within an hour, with the average time being less than 30 minutes. But even benign viruses are almost always damaging, even if this is unintentional. Viruses occupy memory and disk space, and are known to interfere with printing. They can also live at very low levels in the operating system and interfere with other parts of the system, including causing the system to slow down and perhaps crash.

Computer Worms

A worm is a program that replicates itself and spreads, but does not attach itself to other programs. It differs from a virus in that it does not require another program in order to survive. Worms usually spread within a single machine or over a network. They are not spread by sharing infected software. Worms can potentially paralyze a network, as did the famous Fall 1988 Internet worm. This worm infected and disabled several thousand government and university computers running UNIX-based operating systems in a single day.

Keeping an Individual Account Secure

This subsection contains suggestions for keeping individual accounts and local machines secure. Consider putting some or all of the suggestions into effect for your account and machine, depending, of course, on the security requirements of your site. Additionally, if you administer a machine with more than one user, for example, a time-sharing standalone, inform the other users of these precautions.

Safeguarding Initialization Files

Make sure that you, *and only you*, own and have write permissions to your initialization files: `.login`, `.cshrc`, `.profile`, `.mailrc`, `.sunview`, `.exrc`, among others. Also, ensure that only you have write permission for your home directory. Otherwise, someone could modify your `.login` or `.profile` file and get a copy of a shell `setuid` to your user ID, without your ever knowing it. Shells that are `setuid` are very convenient ways to penetrate an account. Shells that are `setuid root` are very convenient ways to penetrate an entire system.

Monitoring the `.rhosts` File

The `.rhosts` file is a mechanism whereby users can permit remote login access to their account to other users. *The SunOS Network Environment* and the `hosts.equiv(5)` man page discuss this file fully. In general, it is a bad idea to give away login access for your account to anybody else. Suppose you get an account on a remote machine that normally requests your password because your machine name is not in its `hosts.equiv` file. Then you can request that the administrator of the remote host put an entry in its `.rhosts` file with your machine name and login account. This is the only safe use of the `.rhosts` mechanism. If you must give away access to your account for a short period, make sure to delete the `.rhosts` file you created as soon as it is no longer needed. In the past few years, many network break-ins have occurred as the result of shoddy `.rhosts` maintenance.

Monitoring Security Through Options to `ls`

The `-ls` command includes three very useful options for monitoring account security.

- `-a`, to display all files.
- `-l`, to display a long format.
- `-c`, to show when the inode was last changed. (Normally the `-l` option shows the time of last modification, which doesn't include mode changes as does `-c`.)

When you set up a new account, and from time to time after that, list your home directory using these options, just to see if anything is amiss.

```
% ls -alc
total 9911
drwxr-x--- 30 tut          1536 May  3 11:00 .
drwxr-xr-x 29 root         512 Apr 17 15:07 ..
-rw-r----- 1 tut          293 Apr 10 14:45 .cshrc
-rw-r----- 1 tut          187 Apr 10 14:18 .login
-rw-r----- 1 tut          769 Apr  1 11:54 .mailrc
-rw-r----- 1 tut          246 Apr 10 11:36 .rootmenu
-rw-r----- 1 tut          425 Feb 17 21:53 .sunview
...
```

These permissions look reasonable for an account with a desired `umask` of 27, enabling members of a user's group to read his or her files. The inode modification times also look good. Make sure you own all files and directories in your home directory except `..` (your home directory's parent directory).

Creating a Safe Search Path

Because of the possibility of Trojan horses, you should never have the current directory as the first element of your search path. This is particularly important for the `root` account. C shell users should place the first line below in `.login`, and Bourne shell users should place the second and third lines in `.profile`:

```
set path = ($HOME/bin /usr/ucb /bin /usr/bin .)
-----
PATH=$HOME/bin:/usr/ucb:/bin:/usr/bin:.
export PATH
```

Placing the current directory last in the search path has the beneficial side-effect of improving efficiency. Just get used to typing `./cat` when you're testing your own `cat` program. Better yet, don't name your programs after system utilities.

Notice that this safe search path isn't the default.

Temporary Directory Awareness

Programs often use the directories `/tmp` and `/var/tmp` to stash temporary files. Users often place files there when they're out of space in their home directory, or when they don't want to bother saving a file. Be aware that the general public has read permission for both `/tmp` directories. Unless files you put there are unreadable, they will be open to everyone. However, only the owner and the superuser can delete a file in `/tmp` because the sticky bit is set. On the other hand, if read permission is granted, everybody can read it.

Protecting Unattended Workstations

Never leave your workstation unattended, unless you can lock your office. Run `lockscreen` to avoid having to exit `sunview`, log out, log in again, and re-enter `sunview`. If you're leaving for vacation, exit `sunview` and log out.

When you run `lockscreen`, somebody can reboot your system, but they can't access your account, because they will just get a `login` prompt after the system reboots. If you remove the word "secure" from the `console` entry in `/etc/ttytab`, they can't reboot single user without the root password.

Keeping File Systems Secure

This subsection contains tips for keeping file systems secure. As system administrator, you must keep special devices unreadable, monitor `setuid` programs, and control the mounting and unmounting of file systems. Instructions for performing these tasks follow.

Ensuring Device Security

The SunOS operating system communicates with attached devices (such as the network, disks, terminals, printers, and modems) by means of special files. All attached devices have associated special files in the `/dev` directory, to which system calls (such as reads and writes) are made.

Treating devices as files allows programs to be device independent: they don't need to know the specifics of the device they're using. The device driver takes care of details such as disk sectoring, network protocols, and buffering. The program simply opens a device file and reads from and writes to it.

This arrangement is also helpful from a security standpoint, since all a device's I/O goes through a small number of channels – the special files. As long as the

special files have the right protection, users cannot access the devices directly.

Note: In this example, the asterisk (*) is used to denote a series of files that have the same initial letters. That is, `rsd*` stands for `rsd0`, `rsd1`, and so on.

Here are special files that are particularly sensitive from a security standpoint.

```
0 crw-r----- 1 root kmem 7, 0 Aug 1 19:28 drum
0 crw-rw---- 1 root kmem 41, 0 Feb 9 19:26 dump
0 crw-r--r-- 1 root staff 3, 11 Aug 1 19:28 eeprom
0 crw----- 1 root staff 16, 0 Feb 9 19:26 klog
0 crw-r----- 1 root kmem 3, 1 Feb 9 19:26 kmem
0 crw----- 1 root staff 3, 4 Feb 9 19:26 mbio
0 crw----- 1 root staff 3, 3 Feb 9 19:26 mbmem
0 crw-r----- 1 root kmem 3, 0 Feb 9 19:26 mem
0 crw----- 1 root staff 37, 40 Feb 9 19:26 nit
0 crw-r----- 1 root operator 17, 0 Feb 9 19:27 rsd*
0 crw-r----- 1 root operator 9, 0 Feb 9 19:27 rxy*
0 brw-r----- 1 root operator 7, 0 Feb 9 19:27 sd*
0 crw----- 2 root staff 3, 5 Feb 9 19:26 vme*
0 brw-r----- 1 root operator 3, 0 Feb 9 19:27 xy*
```

The file `kmem` represents kernel memory, while `mem` represents the entire system memory. These files should be readable by group `kmem` so that the `ps` command can read memory, but they should not be readable by anybody else. If clever enough, a user who can look through memory could read private data. Disk drive interfaces such as `rxy*` and `xy*` (these may have different names on your system) should be readable by group `operator` so that the `df` command can inspect them for free space. No commands need to read the VME bus interfaces `vme*`, the Multibus interfaces `mb*`, or the network interface `tap nit`. (Those that do need to read `nit`, or read/write `eeprom` should be usable by the superuser only, anyway.)

All these sensitive files should have the proper mode and ownership when you install SunOS Release 4.1. Check these files on your system and the display above to make sure.

Controlling `setuid` Programs

Your system should have only a limited number of `setuid` root programs. Spurious programs of this ilk often pose security risks.

You can list some system programs that are `setuid` by issuing this series of commands.

```
% ls -lsg /usr/bin | grep rws
24 -rwsr-xr-x 1 root staff 24576 Oct 15 15:53 at
16 -rwsr-xr-x 1 root staff 16384 Oct 15 15:53 atq
16 -rwsr-xr-x 1 root staff 16384 Oct 15 15:53 atrm
24 -rws--s--x 1 root daemon 24576 Oct 15 16:09 cancel
32 -rwsr-xr-x 3 root staff 32768 Oct 16 09:20 chfn
32 -rwsr-xr-x 3 root staff 32768 Oct 16 09:20 chsh
24 -rwsr-xr-x 1 root staff 24576 Oct 15 15:53 crontab
17 -rwsr-xr-x 1 root bin 16458 Oct 15 15:55 fusage
24 -rwsr-xr-x 1 root staff 24576 Oct 16 09:20 login
32 -rws--s--x 1 root daemon 32768 Oct 15 16:09 lpstat
24 -rwsr-xr-x 1 root staff 24576 Oct 15 15:52 mail
5 -rwsr-xr-x 1 root staff 4936 Oct 15 15:52 newgrp
32 -rwsr-xr-x 3 root staff 32768 Oct 16 09:20 passwd
7 -rwsr-xr-x 1 root staff 6280 Oct 15 16:08 su
64 -rws--x--x 1 uucp daemon 65536 Oct 15 16:07 tip
32 -rwsr-xr-x 1 root staff 32768 Oct 16 09:20 yppasswd
```

The `at*` commands and `crontab` are setuid to root, so that users can place jobs in `/var/spool/cron`, which is writable only by root.

The commands `chfn` (change full name) and `chsh` (change shell) are links to `passwd`, which needs to be setuid root so it can modify the `passwd` database. Likewise, the `mail` command is setuid root because it deals with files and directories that require superuser access.

The `login`, `su`, and `newgrp` commands are setuid root because they have to change user and group IDs when people log in, switch user, or switch group.

The `cu` and `tip` commands (not shown above) are setuid `uucp` because they maintain a lock file in a directory writable only by `uucp`. The modem device these commands use to call out is generally owned by `uucp`, as well.

You can use the `find` command to locate all the setuid root programs on a system. Note that this command takes a long time to execute. You can also use `ncheck -s` as described below. (The `ncheck(8)` man page completely describes this command.) `ncheck` takes less time to execute than `find`, but does not provide as much information. Here is an example using `find` to locate setuid root programs.

```
# find / -user root -perm -4000 -exec ls -lg {} \;
-rwsr-xr-x 1 root staff 5072 Feb 5 12:28 /usr/bin/newgrp
-rwsr-xr-x 1 root staff 24576 Feb 5 12:28 /usr/bin/login
-rwsr-xr-x 1 root staff 24576 Feb 5 12:28 /usr/bin/mail
-rwsr-xr-x 3 root staff 24576 Feb 5 12:28 /usr/bin/passwd
-rwsr-xr-x 1 root staff 16384 Feb 5 12:28 /usr/bin/su
-rwsr-xr-x 3 root staff 24576 Feb 5 12:28 /usr/bin/chsh
-rwsr-xr-x 3 root staff 24576 Feb 5 12:28 /usr/bin/chfn
-rwsr-xr-x 1 root staff 16384 Feb 5 12:42 /usr/bin/crontab
-rwsr-xr-x 1 root staff 24576 Feb 5 12:42 /usr/bin/at
-rwsr-xr-x 1 root staff 16384 Feb 5 12:42 /usr/bin/atq
-rwsr-xr-x 1 root staff 16384 Feb 5 12:42 /usr/bin/atrm
-rws--s--x 1 root daemon 24576 Feb 5 12:40 /usr/ucb/lpr
-rws--s--x 1 root daemon 24576 Feb 5 12:40 /usr/ucb/lpq
-rws--s--x 1 root daemon 24576 Feb 5 12:40 /usr/ucb/lprm
-rwsr-xr-x 1 root staff 16384 Feb 5 12:41 /usr/ucb/quota
-rwsr-xr-x 1 root staff 65536 Feb 5 12:41 /usr/ucb/rcp
-rwsr-x--x 1 root staff 57344 Feb 5 12:40 /usr/ucb/rdist
-rwsr-xr-x 1 root staff 16384 Feb 5 12:41 /usr/ucb/rlogin
-rwsr-xr-x 1 root staff 16384 Feb 5 12:41 /usr/ucb/rsh
-rwsr-sr-x 1 root tty 114688 Feb 5 12:26 /usr/etc/dump
-rwsr-xr-x 1 root staff 98304 Feb 5 12:26 /usr/etc/restore
-rwsr-xr-- 1 root operator 90112 Feb 5 12:26 /usr/etc/shutdown
-rwsr-xr-x 1 root staff 16384 Feb 5 12:43 /usr/etc/keyenvoy
-rwsr-xr-x 1 root staff 16384 Feb 5 12:47 /usr/etc/ping
-r-sr-x--x 1 root staff 114688 Feb 5 12:24 /usr/lib/sendmail
-r-sr-x--x 1 root staff 131072 Feb 5 12:24 /usr/lib/sendmail.mx
-rwsr-xr-x 1 root staff 24576 Feb 5 12:40 /usr/lib/ex*recover
-rwsr-xr-x 1 root staff 16384 Feb 5 12:40 /usr/lib/ex*preserve
-rws--s--x 1 root daemon 57344 Feb 5 12:40 /usr/lib/lpd
```

These programs are fine the way they are. Many programs need to be `setuid` in order to read special system files. Other programs need to be `setuid` to change user and group IDs as needed. Still others need to be `setuid` in order to accomplish administrative tasks.

Note: `ncheck` doesn't work across the NFS; run it on a server.

Aside from `setuid` programs, you should also check periodically that there are no special files outside of `/dev`. Only the superuser can create special files with the `mknod` system call, so NFS user partitions shouldn't contain special files. On the NFS file server, go through user file systems one by one, invoking the `-s` option of `ncheck` to verify that user partitions are free of special files.

```
# ncheck -s /dev/xy2c
1099 /home/raks/tut/floppy
```

This indicates that user `tut` has somehow installed a special device for a floppy drive in his home directory. This would be impossible for him to do unless he had broken security somehow. It could also indicate that the file in question is `setuid`.

You should go through the permissions of the files installed from the release medium. Note any that are run `setuid`, and note who is the owner. Some programs, such as daemons, always need to execute with root as user ID, regardless

of who actually executed them. Depending on your site's security considerations, it is sometimes safer to change the ownership of other programs from root to daemon, if possible.

You should periodically check for new (and possibly dangerous) `setuid` programs in the system. Use `find` in a script regularly called by `crontab` (Refer to the *Administering Workstations* chapter for instructions on using `crontab`).

Shell scripts and programs, if run with root ownership and the `setuid` bit on, can constitute dangerous security holes. To prevent this kind of security problem, you can mount a file hierarchy with the `nosuid` option, as follows.

```
# /usr/etc/mount -o rw,nosuid server:pathname mount_point
```

If you enable this option, then the `setuid` and `setgid` bits will be ignored for any program executed from a hierarchy mounted with the `nosuid` option.

Mounting and Unmounting File Systems

You use the `mount` and `umount` commands to mount file systems, either from devices attached to your local machine or from remote machines over the network. The chapters *Introducing the SunOS Operating System* and *File Maintenance* contain numerous examples of local mounts. The later chapters *The Sun Network File System Service* and *The Remote File Sharing Service* contain examples of remote mounts.

When a new file system is mounted, the original files below the mount point are no longer accessible. Therefore, don't put files in directories you plan to use as mount points. Typical names for mount point directories are `/mnt` and `/arch`. Note that after mounting a file system, the permissions and ownership of the mount point take on those of the root directory of that file system. Here is an example of this effect.

```
# ls -ld /usr/dancetool
drwxr-xr-x  2 root          24 Dec 10  1986 /usr/dancetool
# mount samba:/usr/dancetool /usr/dancetool
# ls -ld /usr/dancetool
drwxr-xr-x 12 tut           512 Jul 21 15:07 /usr/dancetool
```

So be careful when you mount a file system. Make sure that the newly-mounted hierarchy does not grant read and write permission to everyone. Also check that there are no spurious devices or `setuid` programs on newly mounted file systems.

Much of the time it is a good idea to mount file systems read-only, as explained in *File Maintenance*, *The Sun Network File System Service*, and *The Remote File Sharing Service*, and the `mount(8)` man page. This solves the problem of tampering with writable files and directories, but not the problem of bogus `setuid` programs. Use the `nosuid` option of `mount` to prevent problems. When mounting an unfamiliar file system, run the `ncheck -s` command to verify that everything is as it should be.

Keeping System Directories and Files Secure

System directories should be owned by `root` and grant full permissions to the owner, but deny write permission to group and others—755 in absolute mode. If a system directory is writable, a cracker could move files around and install new programs, possibly Trojan horses. Here are the important system directories.

```

/                /usr/etc
/dev             /usr/lib
/etc            /usr/bin
/usr           /usr/spool
/var          /usr/kvm
/etc/security (if C2 is installed)

```

Likewise, many system files in these directories should be owned by `root` and mode 755 or 644 —write permission denied to group and other. If a program or system file is writable, a cracker could modify these files to break security. For example, if `/etc/passwd` were writable, a cracker could remove the `root` password and become superuser without a password. Likewise, if `/etc` were writable, a cracker could replace your good version of `passwd` with a bad version. Here are some important system files.

```

/vmunix         /etc/rc
/usr/bin/*      /etc/fstab
/usr/ucb/*      /etc/passwd
/usr/local/*    /etc/group
/dev/*mem       /var/spool/cron/crontabs/root

```

To find all directories on your system that everyone has permission to write and execute, run this command.

```

# find / -type d -perm -777 -print
/etc/sm
/etc/sm.bak
/tmp
/var/spool/mail
/var/spool/uucppublic
/var/spool/secretmail
/var/tmp

```

The directories `/etc/sm` and `/etc/sm.bak` are used by the status daemon `statd(8c)` and the lock daemon `lockd(8c)`. Traditionally, everyone has write permission for the directory `/tmp`. But for enhanced security, the sticky bit is set, changing its mode to 1777. This prevents anyone but the owner and the superuser from removing a file.

The sticky bit is also set for the mail spooling directory `/var/spool/mail` but not for the secret mail directory `/var/spool/secretmail`. If you use `secretmail` at your site, change this directory's mode. The directory `/var/spool/uucppublic` is also traditionally left wide open, but with the sticky bit set.

Protecting the `/etc/passwd` File

From the standpoint of security, `/etc/passwd` is the most important system file. Verify that it is mode 644, owned by `root`, and that the directory `/etc` is owned by `bin` and its mode is 2755. Also verify that there are no empty password fields in the file. You can do this by running the following command:

```
% awk -F: '$2 == "" {print}' /etc/passwd
sync::1:1:1:/:/bin/sync
+::0:0:0:::
```

The only output lines should be the entry for `sync`, which appears on all systems and the line

```
+::0:0:0:::
```

which appears for systems on a network running NIS. It's also very important that no two users have the same user ID.

If your systems are configured C2 secure, make sure that all lines in `/etc/passwd` contain a `##` in the second field, rather than an encrypted password. Here is how you can verify this.

```
% grep -v "##" /etc/passwd
+::0:0:0:::
```

The only output line should be the one shown. The `-v` flag of `grep` says to print only the lines that don't match the pattern.

Administering Password Aging

In a secure environment, an important security practice is to change passwords on the local machine on a regular basis. By implementing *password aging*, you can ensure that users change their passwords frequently.

The password aging process enables you to specify the time allotted for a password to "mature." When the password's maturity date has elapsed, the `login` program will require users to change their passwords. You also can specify the minimum interval of time that must pass before a user can change his or her password at will.

You might want to set up password aging on client workstations and servers if their network requires a very secure environment. Also consider setting up password aging on non-networked standalones and time-sharing systems that handle very sensitive data. You set up password aging on each local machine, on a user-by-user basis. Network-wide password databases provided by NIS do not support password aging.

The `/usr/bin/passwd` command includes options that enable aging. You must be superuser or the user whose password will be aged in order to run these options. You can also set up password aging for the root password, to ensure that you will change it.

Setting a Password's Maximum Age

The first step in setting up password aging is specifying the maximum number of days that the password can age before it expires. When the interval is up, the user must change the password. You use the `-x` option of `passwd` to do this, as

shown in the following syntax:

```
/usr/bin/passwd -x maxdays user_name
```

Here *maxdays* is the maximum number of days that this password is valid, after which `login` will require user *user_name* to change the password.

Suppose you decide to implement password aging for user *stefania* as follows.

```
# passwd -x 60 stefania
```

Now user *stefania*'s password has approximately two months to mature. If *stefania* has not changed her password and the two months elapses, she will receive the following message the next time she logs in to her machine.

```
Your password has expired. Choose a new one.
```

`login` then runs the familiar `passwd` command and prompts *stefania* to provide the old password and the new password.

Set up a reasonably long interval before passwords mature, so that users don't to change their passwords too frequently. Frequent password changes may force users to pick passwords that are easy to remember, since the passwords must change constantly. Passwords that are easy to remember are also easy for a would-be cracker to guess.

Setting a Password's Minimum Age

Note You must set up a password's maximum age before specifying its minimum age.

Some sites may not want users to change their passwords before an interval of time has elapsed. You use the `-n` option of `passwd` to specify a password's minimum age.

Note that you can set the minimum number of days only if password aging is already in affect. Use the following syntax.

```
/usr/bin/passwd -n mindays user_name
```

Here *mindays* is the absolute minimum number of days that must pass before user *user_name* can change his or her password.

To set the minimum age of user *stefania*'s current password, you can type the following.

```
# passwd -n 3 stefania
```

If *stefania* tries to change her password before three weeks elapse, she will get the message:

```
Sorry, less than 3 weeks since the last change
```

However, suppose someone discovers *stefania*'s password before it reaches its minimum age, and she needs to change it. Therefore, you as superuser must

issue the following and have stefania type in her new password as prompted.

```
# passwd stefania
```

Immediately Expiring Passwords

In rare cases, a site may be compromised by unauthorized intruders. Should this happen, you can force passwords to expire immediately. Thereafter, users have to change their passwords the next time they log in. Use the following syntax of `passwd`.

```
/usr/bin/passwd -e user_name
```

Do this for each user at your site. The next time the users log in, the `login` program will run the `passwd` command, forcing the users to change their passwords at that time.

Displaying Password Aging Information

You use the `-d` option of `passwd` to display password aging information for a particular user. Here is the syntax.

```
/usr/bin/passwd -d user_name
```

You must be superuser or logged in as `user_name` above to use the `-d` option.

Suppose you want to view password information for user `stefania`. You would type the following.

```
# passwd -d stefania
```

Here is the resulting display.

```
stefania 11-17-89 14 60
```

To display password aging information for all accounts on the local machine, you must become superuser, then use the following syntax of `passwd`:

```
# /usr/bin/passwd -d -a
```

The resulting display will resemble the following for machine `raks`.

```

root
nobody
daemon
sys
bin
uucp
news
ingres
audit
sync
sysdiag
stefania 11-17-89 14 60
azhar
shamira

```

Note that only user stefania has password aging implemented.

Protecting the /etc/group File

Like the password file, the group file should be mode 644, owned by root. An empty password field in this file means that only people mentioned in the user list field can gain access to that group by using the `newgrp` command. If the password field is not empty, anybody knowing the password can gain access to the group by using the `newgrp` command.

Group passwords are rarely used. By convention, a star (*) is used to fill up the field. Here is a command to check for empty group passwords.

```
% awk -F: '$2 == "" {print}' /etc/group
+:
```

The only output line should be the one shown, which includes group entries from NIS.

Protecting the /etc/aliases File

Like the `passwd` and `group` files, the `sendmail /etc/aliases` file must have an absolute mode of 644, owned by root. (See the *Administering Electronic Mail* chapter for more information about the `/etc/aliases` file.)

Protecting /var/spool/cron

The `cron` daemon executes at the appropriate times the commands listed in the corresponding `/var/spool/cron/crontabs/*` file for a given user. You should verify that the file `/var/spool/cron/crontabs/root` contains no security holes. Do not execute any command or shell scripts that are writable by anybody but root, or that are in directories writable by anyone other than root or bin. Otherwise somebody could replace these programs, and thereby gain superuser access. Most other files in the `crontabs` subdirectory do not pose a security risk. However, if any other account exists on the system with any special privileges whatsoever, such as `uucp` account, then you should keep the `crontab` files for those accounts secure. This is true for all “system” accounts, such as `bin`, `daemon`, `uucp`, and so on.

Promoting Physical Security and User Awareness

Physical security is a machine's first line of defense, but is perhaps not as important as the human element. Here are suggestions for promoting the physical security of computer systems.

- Keep the computer in a locked room or building with a good key-card access system. Very secure environments might install alarms and hire guards.
- Install a good fire prevention system and implement plans for backup and recovery systems.
- Shield the computer room and all cabling with copper, so that RF signals can't be detected outside.
- Do not install any communications lines outside the secure area. If you must, employ encryption devices and other security mechanisms.
- Keep sensitive output in locked boxes. Use locked trash bins and shredders for discarded output.

The guiding principle of physical security is that security measures should not cost more than your computer system and its data are worth. Paying \$150,000 a year for armed guards to protect a \$7,000 workstation is not worth it unless the workstation contains important data. On the other hand, \$500 for a paper shredder (or burner) to destroy listings of top secret programs is money well spent.

Communications lines are the most vulnerable aspect of physical security. As soon as you install a dial-up phone line, your system is potentially open to anybody who might call. Even an unlisted number won't help much, because there are home computers programmed to dial numbers in sequence. Using a call-back modem or a local PBX might help, since outside callers must dial an extra extension number. DES encryption of all outside communications is possible, but requires special-purpose hardware and software.

Promoting User Awareness

As system administrator, it is your duty to make users aware of security issues. Encourage users to read the sections of this chapter that deal with protecting the local machine and file protection. When you spot a security problem, talk to the responsible user in an effort to reform lax security practices. Most of the time bad security is the result of ignorance rather than impudence.

A milligram of prevention is worth a gram of cure. When you install new accounts, give every user a `.login` file that contains these lines.

```
umask 27
set path = (~ /bin /usr/ucb /usr/bin /usr/local .)
```

The actual path in the second line may be different, but the dot indicating the current directory is strongly recommended. The first line makes sure that by default, files created by that user cannot be written to by any others. The members of the user's group can read and execute his or her files, but not the public.

Send mail to users if they have files that grant write permission to the public. They might not have intended this. Also send mail if users have `setuid` programs in their directories. Periodic mail about security will get users to think about the subject.

Management awareness is important. Security policies are much easier to enforce if management is on your side. Try to co-opt management by having them come up with a set of security guidelines. Security standards will be easy to enforce this way. Management can help by impressing upon users that electronic information is a form of property.

Educate users as much as possible. Teach them how permissions work. Instruct them on how to choose a good password, by showing them the password discussion in *SunOS User's Guide: Doing More*. Encourage them to change passwords periodically, and, if necessary, implement password aging. The most-secure user population is a well-educated one.

Administrator Awareness

Everything stated above applies to you as system administrators as well, and even more so. If a user's login is compromised, only that user's files (and perhaps files in that group) are compromised. But if the `root` login is compromised, the entire system, and probably the entire network, are compromised. In fact, the wide powers of the `root` login make it the most inviting target for security attacks.

Suggestions for Keeping Systems Secure

Here is a summary of steps you can take to keep your systems secure.

- Think about your system's vulnerabilities. If you have modems, are they attached to dial-in lines, and are the phone numbers published? If you are attached to a wide-area network, who else is attached to it? Do you use programs from unknown or unreliable sources? Do you have sensitive information on your machine? Are your users knowledgeable about security issues? Is management committed to security?
- Guard the integrity of your file systems. Check the permission of system files periodically to make sure they aren't writable by anybody except the superuser. Make sure you have no bogus `setuid` or `setgid` files lying around, especially ones owned by system users and groups. Be careful about permissions on your device files. Don't mount file systems without checking them first.
- Keep disk backups in a secure, fire-resistant area. From time to time, transfer a full dump tape to a fireproof vault.
- Keep track of your users and which systems they are authorized to use. Find stale logins and disable them. Make sure there are no accounts without passwords.
- Use the auditing features described in *Administering C2 Security*. Look for unusual usage patterns: huge disk consumption, large amounts of CPU time, many processes, lots of invalid superuser or login attempts, and concentrations of network traffic to a particular system.

- When installing software from an unknown or unreliable source, check the source code for peculiar attempts to create `setuid` files or tamper with the password file. Even when installing software from a reliable source, check `setuid` and `setgid` programs to make sure this is really necessary. Many programs that set the user ID to `root` could easily be set to an alternate login ID.

What if Security is Broken?

When regular security checks or evidence from the audit trail indicates that system security has been compromised, you should take immediate action.

If the security attack came from within, you should confront the offending user and ask what happened. It is possible that the user just made a mistake. If the security breach was not malicious, it may be a simple matter of educating the user and paying close attention to him or her in the future. If damage was done to the system, you should report the incident to management and to users whose files were affected.

If you cannot identify the source of a security attack, you should assume the worst. If a capable cracker managed to become `root`, your system files and programs are compromised to an unknown degree. You should attempt to find out who that person is and what damage has been done, if possible. But first, take the following action.

1. Shut down the system with `shutdown`, and enter single-user mode. Do not return to multiuser mode.
2. Mount the `/usr` file system and copy the programs from `/usr/bin` and `/usr/lib` to a temporary directory.
3. Mount the tape that contains the original distribution of the operating system, and reinstall the system as you did in the beginning.
4. Compare the versions of `/usr/bin` and `/usr/lib` that you copied to a temporary directory with those you just installed. If they are different, you can be sure system integrity was broken.
5. Mount users' home directories and run `find` and `ncheck` to make sure there aren't any new Trojan horses in these file systems.
6. Change all passwords on the system. Inform users that their passwords are changed and that they should contact you for a new password.
7. When you give them a new password, tell users that there has been a security breach. Ask them to check their accounts for anything unusual.
8. Try to determine how the break-in occurred. This may be impossible without talking to the person who broke in. Many companies pay system crackers in exchange for advice on how to plug security holes.

Any security break results from either inadequate physical security or from the human factor. If inadequate physical security was your problem, improve it. It is more likely that the problem was caused by the human element. Increased user education and administrator vigilance are the solutions here.

Administering Workstations

This chapter discusses typical tasks that you perform to administer an individual workstation, and how to add workstations and software to an already running network. Topics include:

- Setting up the operating environment on local machines.
- Adding client workstations to already functioning NFS file servers.
- Adding support for additional kernel and application architectures to an NFS file server.
- Adding additional software to already running machines.
- Handling power loss problems and system crashes.
- Scheduling maintenance programs through `crontab` files.
- Setting up the system log configuration and system performance tools.
- Setting up system accounting.

8.1. Setting Up a Workstation's Operating Environment

You have to tailor default administrative databases for individual workstations whenever you:

- Install a new SunOS release through SunInstall.
- Add a new workstation to an existing network.

This section explains how to update these databases for any Sun computer with a new SunOS release. You will need to perform these tasks, regardless of whether you administer a file server, a time-sharing system, a standalone, or just your own diskless or dataless client.

Customizing the Administrative Databases

The next instructions assume that you, or the administrator of your network, have installed the SunOS Release 4.1. The steps show how to tailor the administrative databases first introduced in *Introducing the SunOS Operating System*, so that your machine can operate properly. (The later section, *Adding Diskless Workstations and Software*, explains how to expand your configuration by adding these items.) Unless otherwise noted, you must be superuser to perform these procedures.

Step 1: Creating Home Directories

The following steps show how to create a home directory on a file server, a networked machine with a disk, a diskless client, and a non-networked client. You may need to create a home directory for yourself or each user on a diskless client in your server's /home file system. You may also need to create home directories for users on standalones and for dataless clients who request you to do so.

For a file server:

Go to the /home file system and do the following:

```
# cd /home
# mkdir server_name
```

Now type the following for every user you support:

```
# cd server_name
# mkdir user_name
```

For other networked machines with disks:

If you want your home directory to be on a local disk, create it as follows:

```
# cd /home
# mkdir -p machine_name/user_name
```

If your system disk is under 110 MBs, you have two default mount points, / and /usr. You should create your home directory under /usr/export/home. /home is a symbolic link to /usr/export/home. If you have other mount points, then the link is reversed /home is created, and /usr/export/home is a symbolic link to /home.

For diskless clients:

The administrator of your NFS file server must create your home directory for you. Notify the administrator if after a certain period of time your home directory was not created.

For non-networked machines:

Create a home directory under /home by typing:

```
# cd /home
# mkdir -p machine_name/user_name
```

If your machine is a time-sharing system, repeat this step for every user that will have an account on your system.

Step 2: Set Up Password Security

To insure security of an individual machine, you must update its local `passwd` database, and create passwords for your personal user name and root. The next instructions apply to all types of Sun computers.

When you first log in to a newly installed machine, you should log in as root, with no password. As root, do the following:

1. Edit your local `/etc/passwd` file using your preferred text editor.
2. Add a password entry for yourself and any other users who will locally log in to your machine, as explained in the chapter *The Sun Network File System Service*.
3. Leave the actual password field blank. (The second field, separated by ":" on each line.)
4. Fill in the home directory path and preferred login shell for everyone who will log in to your machine, then save and exit the file.
5. Run the `passwd` command and create a password for your user login name as follows:

```
# passwd user_name
# New password: your password
# Retype new password: your password
```

6. Create a root password for yourself as follows:

```
# passwd root
# New password: your password
# Retype new password: your password
```

Step 3: Setting Up File Sharing

This next step is appropriate only for administrators of file servers and their clients. It does not apply to machines that are not on a network.

In order for clients to receive their files from the appropriate file server, the following must happen:

- Administrators on the file servers must export (through NFS) or advertise (through RFS) the directory hierarchies that they have available.
- Users or administrators on network clients must create mount points for these directories, then add entries to their local `/etc/fstab` files for each directory hierarchy they want to mount.

The Sun Network File System Service and *Using the NFS Automounter* explain how to set up NFS file sharing. *The Remote File Sharing Service* explains RFS file sharing.

Step 4: Setting Up User Groups

Perform the following steps if user groups will exist at your site. The chapter *Administering Security* describes the `/etc/group` file. *The Sun Network File System Service* also has more information for establishing user groups within a networked environment.

On a Network with NIS:

If you are the administrator in charge of the NIS domain, create the network groups as needed, then add them to the `netgroup` database, as explained in the *The Network Information Service* chapter.

On a Network Without NIS:

Each client needs a copy of the NFS file server's version of `/etc/group`. If you are the administrator of the file server, add groups as they are created to your copy of `/etc/group`. Then copy this file into the `/export/root/client_name/etc` file for every client you support.

For Non-Networked Time-Sharing Systems:

Add groups to `/etc/group` as you need them.

Step 5: Enabling Printing

Machines involved in printing are the *print server*, which is the machine with a locally attached printer, and the print server's clients. Any machine with serial ports and a disk can become a print server, including non-networked standalones and time-sharing systems.

On the Print Server:

Set up your printer hardware by following the instructions from the printer manufacturer. Edit your default `/etc/printcap` file, and create spooling directories on your machine, as described in the chapter *Adding Hardware to Your System*.

On Network Clients

Edit your default `/etc/printcap` file so that it includes entries for every printer on your network. Refer to *Adding Hardware to Your System* for more information.

Step 6: Enabling Electronic Mail

Electronic mail can be set up for both a network and a non-networked time-sharing machine. The chapter *Administering Electronic Mail* explains how to do this.

Step 7: Ensuring Network Security

On all machines on a network, edit your local `/etc/hosts.equiv` and `.rhosts` files to specify what machines are allowed to remotely log in to your machine. (See *The SunOS Network Environment* for more information about `hosts.equiv` and `.rhosts`.) If your network requires more security, refer to the added security features described in *Administering Security* and in *Administering C2 Security*.

8.2. Adding Diskless Workstations and Software

An important aspect of your job as a system administrator is to add new workstations to your network. The major tasks to perform in this area are:

- Adding and removing NFS clients to and from existing servers.
- Converting a standalone system to a server.
- Adding additional software categories to a server or a standalone from a release tape.

This section assumes you are familiar with basic concepts like NFS and NIS. (See *The Sun Network File System Service* and *The Network Information Service* for more information.)

You use `add_client` to add an NFS client to an existing server. Before adding a new client, make sure that the IP and Ethernet addresses for that client are listed in the `hosts` and the `ethers` database, either NIS maps on a network running NIS, or in the server's local `/etc/hosts` and `/etc/ethers` files. If `add_client` does not find the client entry in the `hosts` database, or in the NIS `hosts` map, it aborts without adding the client.

In order to operate, a client must have access to the file systems containing the kernel, major commands, and executables. Using `add_client` automatically creates the `/export/swap/client_name` and `/export/root/client_name` files in the server's `/etc/exports` file. If you are adding a client to a server, `add_client` also updates the following files:

```
/etc/hosts
/etc/exports
/etc/ethers
/etc/bootparams
```

If your server is a NIS master, `add_client` also does a `make` in `/var/yp`.

NOTE *When adding a client, take into consideration your disk size. Consider that the client's root needs about 5MB, and the client's swap needs about 16MB.*

Using the `add_client` Utility

`add_client` is found in `/usr/etc/install`, and you must be in that directory to use the utility. You must be superuser to use `add_client`. The syntax for `add_client` is as follows:

```
# add_client [-options] clientnames
```

The italicized arguments on the command line are described as follows.

options refers to a series of options you can apply to `add_client`. The options you commonly use are described below. Refer to the `add_client(8)` man page for a complete description of these options.

a *arch* The client's kernel architecture (for example `sun4c`, `sun3x`) if it is different from the server.

- e** *exec-path* Sets the pathname of the executables directory for the application architecture specified. The default is `/export/exec/appl_arch.os.rel`. (This directory should not be changed.)
- f** *share-path* Sets the pathname of the `share` directory, which is usually a link to `/usr/share`. The default is `/export/share/os.rel`.
- h** *home-path* Sets the pathname of the client's home directory. The default is `/home/server_name`.
- i** Interactive mode. Brings up a menu interface.
- k** *kvm-path* Sets the path to kernel executables. (This should not be changed.)
- m** *mail-path* Sets the pathname of the client's mail directory. The default is `/var/spool/mail`.
- p** [*client*] Prints client information giving the client name, release version, architecture, IP number, and root location. Omitting *client* will print the information for all clients.
- r** *root-path* Sets the pathname of the directory for the client root directory. The default is `/export/root/client_name`.
- s** *swap-path* Sets the pathname of the directory for client swap files. The default is `/export/swap/client_name`.
- t** *terminal type* Puts the client's terminal type into `/etc/ttytab`. (The default is `sun`.)
- y** *yptype* Shows the type of NIS server. You can designate "client" or "none". The default is `client`.
- z** *swapsize* Reserves bytes for the client's swap file. *Swapsize* can be kilobytes, blocks, or megabytes. Use **K**, **B**, or **M**, suffixes, respectively. The default is 16M.
- n** Prints out what the settings will be for the client without actually creating the client.
- v** Verbose. Reports the steps as they are performed.
- pv** [*clients*] Prints verbose information on the clients specified. If you do not specify a client, all clients will be shown.

To get a list of the above options, and a brief description, enter `add_client`, and the screen will list them as follows:

```

# add_client
usage:  add_client [[options] clients]
      where 'clients' is the name of the any number of clients to add and
      [options] are one or more of the following for each client:
      -a arch          architecture type (e.g. sun3, sun4, etc.)
      -e path          path to client's executables
      -f path          path to client's share location
      -h path          path to client's home directory
      -i              interactive mode - invoke full-screen mode
      -k path          path to client's kernel executables
      -m path          path to client's mail
      -p              print information of existing client
      -r path          path to client's root
      -s path          path to client's swap
      -t termtype      terminal to be used as console on client
      -v              verbose mode - reports progress while running
      -y type          client's NIS type (client, or none)
      -z size          size of swap file (e.g. 16M, 16000K, 32768b etc.)
      -n              prints parameter settings and exits w/o adding
client

```

Unless you have a specific reason, it is a good idea not to change the defaults described above, as changing them could lead to confusion.

Although you can use `add_client` and its features from the command line as shown above, it is recommended that you use the interactive mode (menus) as you are probably familiar with the menus from SunInstall™.

add_client Examples

You can use the `add_client` utility with the command line options as described above, or you can use `add_client` in interactive mode. The following examples show you how to do both.

This example shows how to add the client *patch* to a server.

1. Enter as superuser in the `/usr/etc/install` directory:

```
# add_client abc
```

The system now prompts you for the correct release number and architecture if the server serves more than one architecture.

2. Enter **y** for the correct choice.

```
Is 'sunos 4.1 sun3' the correct release to use for 'abc' ? (y/n) y
```

Then the system gives you the following messages:

Creating client 'abc'

Updating the server's databases
 Setting up server file system to support client
 Making client's swap file
 Setting up client's file system
 Copying /export/exec/proto.root.sunos.4.1 to /export/root/abc
 Copying binaries

Updating client's databases
 Setting up Client's ttytab

You must now ask the system administrator to update the NIS master's /etc/hosts, /etc/bootparams and /etc/ethers files, because of the change in status of the client 'abc.'

3. If you wish, enter the `-pv` option to `add_client` from the command line to get a long version on the status of all your clients as follows:

```
# add_client -pv
client name           : abc
client architecture  [-a] : sunos 4.1 sun4c
client inet (ip) address : 129.144.114.31
client ethernet address :
client domainname    : happy.corp.com
client exec path      [-e] : /usr
client share path     [-f] : /usr/share
client home path      [-h] : /home/patch
client kvmpath        [-k] : /usr/kvm
client mail path      [-m] : /var/spool/mail
client root path      [-r] : /export/root/abc
client swap path      [-s] : /export/swap/abc
client terminal type  [-t] : sun
client NIS type       [-y] : client
client swap size      [-z] : 16M bytes
```

'abc' has already been created

The next example shows how to use `add_client` with the interactive feature to add a client to a server with the same architecture. If you add a client to a standalone, `add_client` will automatically convert it to a server. If your system is a server, it just adds the client.

1. Enter `add_client` with the interactive option:

```
# add_client -i
```

The following message appears:

Interactive mode uses no command line arguments

- After the above message, the next menu appears. Enter three choices: **Architecture Type** :, **Client name** :, and **Choice** : at the top of the page. Then, immediately after selecting create, the rest of the screen appears with the client information. If you need to change anything in the client information, like the size of the swap space, you can do so at this time.

```

CLIENT FORM                                [?=help] [DEL=erase one char] [RET=end of input
data]

-----
Architecture Type : x[sun4c.sunos.4.1]
Client name      : abc
Choice          : x[create] [delete] [display] [edit]
Root fs : /export/root (sd3h)          118805504   Hog : sd3h 118805504
Swap fs : /export/swap (sd3h)         118805504   Hog : sd3h 118805504

Client Information :
  Internet Address      : 129.144.114.31
  Ethernet Address     : 8:0:20:1:00:00
  NIS Type              : [none] x[client]
  Domain name          : happy.corp.com
  Swap size (e.g. 8B, 8K, 8M) : 16M
  Path to Root         : /export/root/abc
  Path to Swap         : /export/swap/abc
  Path to Executables  : /usr
  Path to Kernel Executables : /usr/kvm
  Path to Home         : /home/patch
  Terminal type       : sun

Ok to use these values [y/n] y
[x/X=select choice] [space=next choice] [^B/^P=backward] [^F/^N=forward]

```

A screen now appears to tell you what client you set up to be created.

```
CLIENT FORM          [?=help] [DEL=erase one char] [RET=end of input data]
-----
Architecture Type : x[sun4c.sunos.4.1]
Client name       : abc
Choice            : x[create] [delete] [display] [edit]

sun4c.sunos.4.1 Clients:

abc

Are you finished with this form [y/n] y
[x/X=select choice] [space=next choice] [^B/^P=backward] [^F/^N=forward]
```

After entering **y** to show you are done with the form, `add_client` will add all the clients that you entered in the client form. Then it will display the following messages:

```
Updating the server's databases
Setting up server file system to support client
Making client's swap file
Setting up client's file system
Copying /export/exec/proto.root.sunos.4.1 to /export/root/abc
Copying binaries

Updating client's databases
Setting up Client's ttytab

You must now ask the system administrator to update the NIS
master's /etc/hosts, /etc/bootparams and /etc/ethers files,
because of the change in status of the client 'abc.'
```

The last message in the above screen is true if your server is not an NIS server. If your server is a NIS master, the files were updated.

Using the `add_services` Utility

You use `add_services` to set up a system as a server for another architecture or to add additional software categories from a release tape.

Note: You must be superuser to use `add_services` and be in the `/usr/etc/install` directory.

`add_services` also updates the `/etc/exports` file to export the executable directories it installed. After running `add_services`, you should check the `/etc/exports` file to make sure that the new services are exported correctly. See *The Sun Network File System Service* and the `exports(5)` man page for more information.

add_services Examples

You can use the `add_services` utility to add a new release to a system or edit an existing release.

The following example shows how to add a new architecture. You would add a new architecture when changing a homogeneous server to a heterogeneous server.

1. Load the release tape and enter the following:

```
# add_services
```

2. The following menu appears. At this time, select the **[add new release]** category. The message at the bottom of the screen now prompts you to load the release tape in case you have not already done so.

```
SOFTWARE FORM          [?=help] [DEL=erase one char] [RET=end of input data]
-----
Software Architecture Operations :
  x[add new release]  [edit existing release]

Architecture types :
  [sun4c.sunos.4.1]

Media Information :
  Media Device   : [st0] [st1] [st2] [st_] [xt0] [mt0] [fd0] [sr0]
  Media Location : [local] [remote]

Please insert the release media that you are going to install
Press <return> to continue ...
```

Right after you select **[add new release]**, and press **[return]**, the menu appears for you to choose the media information. In this case, you are loading a new architecture from a local CD ROM. The **Media Device** : field now provides a choice, **[st_]**, to allow you to specify a media device that is supported, but not otherwise designated. You can enter 0, 1, or 2.

```
SOFTWARE FORM                [?=help] [DEL=erase one char] [RET=end of input data]
-----
Software Architecture Operations :
  x[add new release] [edit existing release]

Architecture types :
  [sun4c.sunos.4.1]

Media Information :
  Media Device   : [st0] [st1] [st2] [st_] [xt0] [mt0] [fd0] x[sr0]
  Media Location : x[local] [remote]

Ok to use these values to read the table of contents [y/n] y
[x/X=select choice] [space=next choice] [^B/^P=backward] [^F/^N=forward]
```

3. Select the architecture you wish to add to your server, in this case, sun4. To exit the form, after choosing the architecture, type **x** next to the **[exit architecture menu]**.

```
ARCHITECTURE FORM                [?=help]
-----

Please Select An Architecture :

  [sun3.sunos.4.1]
  [sun3x.sunos.4.1]
+  [sun4.sunos.4.1]
  [sun4c.sunos.4.1]

  [exit architecture menu]

[RET/SPACE=next choice] [x/X=select choice] [^B/^P=backward] [^F/^N=forward]
```

4. The menu now shows the categories you selected and gives a message at the bottom of the screen telling you what you loaded.

```
SOFTWARE FORM          [?=help] [DEL=erase one char] [RET=end of input data]
-----
Software Architecture Operations :
  x[add new release]  [edit existing release]

Architecture types :
  [sun4c.sunos.4.1]

Media Information :
  Media Device   : [st0] [st1] [st2] [st_] [xt0] [mt0] [fd0] x[sr0]
  Media Location : x[local] [remote]

You have the sunos 4.1 sun4 media loaded.  Is this ok? [y/n] y
[x/X=select choice] [space=next choice] [^B/^P=backward] [^F/^N=forward]
```

5. You are now ready to select software categories for you new architecture. The next menu shows what values you have before you go on to **Software Categories**.

```
SOFTWARE FORM          [?=help] [DEL=erase one char] [RET=end of input data]
-----
Software Architecture Operations :
  x[add new release]  [edit existing release]

Architecture types :
  [sun4c.sunos.4.1]

Media Information :
  Media Device   : [st0] [st1] [st2] [st_] [xt0] [mt0] [fd0] x[sr0]
  Media Location : x[local] [remote]

Choice : [all] [default] [required] x[own choice]
Executables path : /usr
Kernel executables path : /export/exec/kvm/sun4.sunos.4.1

Ok to use these values to select Software Categories [y/n] y
[x/X=select choice] [space=next choice] [^B/^P=backward] [^F/^N=forward]
```

6. Now you select under **Choice** : what software you want. In this case, **[own choice]**. You are now shown the first software category **root**, told that it is required, and given the **status**, in this case, **not selected**. Since it is not selected, enter **y** in the **Select this media file [y/n] ?** field to select it, and continue with your software selections. The screens are the same for all your software selections.

```

SOFTWARE FORM          [?=help] [DEL=erase one char] [RET=end of input data]
-----
Software Architecture Operations :
  x[add new release] [edit existing release]

Architecture types :
  [sun4c.sunos.4.1]

Media Information :
  Media Device   : [st0] [st1] [st2] [st_] [xt0] [mt0] [fd0] x[sr0]
  Media Location : x[local] [remote]

Choice : [all] [default] [required] x[own choice]
  Executables path : /usr
  Kernel executables path : /export/exec/kvm/sun4.sunos.4.1

Destination fs : / (sd0g)
Name : root (required)
Hog : sd0g
Size : 319014

Select this media file [y/n] y          status: not selected
[x/X=select choice] [space=next choice] [^B/^P=backward] [^F/^N=forward]

```

And the next software category, and so on:

```

SOFTWARE FORM          [?=help] [DEL=erase one char] [RET=end of input data]
-----
Software Architecture Operations :
  x[add new release] [edit existing release]

Architecture types :
  [sun4c.sunos.4.1pre]

Media Information :
  Media Device   : [st0] [st1] [st2] [st_] [xt0] [mt0] [fd0] x[sr0]
  Media Location : x[local] [remote]

Choice : [all] [default] [required] x[own choice]
  Executables path : /usr
  Kernel executables path : /export/exec/kvm/sun4.sunos.4.1

Destination fs : /usr (sd0g)                                34399706
Name : usr (required)                                       Hog : sd0g 34399706
                                                         Size : 12493586

Select this media file [y/n] y                               status: not loaded
[x/X=select choice] [space=next choice] [^B/^P=backward] [^F/^N=forward]

```

7. After selecting all the software you need, the final menu shows you what you have and queries you **Ok to use this architecture configuration [y/n] ?**. If this is the software that you want, hit **y**, and the system will load the software. Otherwise you can scroll back through the menus to select something you missed and need.

```
SOFTWARE FORM          [?=help] [DEL=erase one char] [RET=end of input data]
-----
Software Architecture Operations :
  x[add new release] [edit existing release]

Architecture types :
  [sun4c.sunos.4.1]

Media Information :
  Media Device   : [st0] [st1] [st2] [st_] [xt0] [mt0] [fd0] x[sr0]
  Media Location : x[local] [remote]

Choice : [all] [default] [required] x[own choice]
  Executables path : /usr
  Kernel executables path : /export/exec/kvm/sun4.sunos.4.1

Media Filenames:
  root
  usr (loaded)
  Kvm
  Install (loaded)
  Networking (loaded)
  Debugging (loaded)
  SunView_Users (loaded)

Ok to use this architecture configuration [y/n] y
  [x/X=select choice] [space=next choice] [^B/^P=backward] [^F/^N=forward]
```

8. After the software is loaded, you will see the following menu:

```
Setting up server file system for services
Extracting the sunos 4.1 sun4 'Kvm' media file.
Setting up tftpboot files

Updating bootparams
Updating server's exports file
Configuring inetd for tftpd operation

Starting daemons
Exporting file systems
```

It will take several minutes for the software that just selected to be read from the tape to the disk.

Using rm_clientUse `rm_client` on the command line as follows:

```
# rm_client patch
```

The screen now shows the following output and prompts you for a response to make sure you actually want to remove the *patch* from the server *story*.

```
removing story's /etc/hosts entry...
Remove /etc/hosts entry -> 129.144.52.22      patch
(y/n)? y
removing story's /etc/ethers entry...
Remove /etc/ethers entry -> 8:0:20:1:41:49    patch
(y/n)? y
removing /tftpboot link 81903416...
removing patch's bootparams entry...
Remove client's root directory /export/root/patch (y/n)? y
removing /export/root/patch...
removing /export/swap/patch...
removing database entries for patch...
```

You must now ask the system administrator to update the NIS master's `/etc/hosts`, `/etc/bootparams` and `/etc/ethers` files, because of the change in status of the client 'abc.'

```
client patch removed.
```

The **y** option, forces yes the system prompts so you don't have to do it manually. Entering

```
# rm_client -y abc
```

produces the following:

```
removing abc's /etc/hosts entry...
rm_client: could not open file /etc/ethers
removing /tftpboot link 8190721F...
removing abc's bootparams entry...
removing /export/root/abc...
removing /export/swap/abc...
removing database entries for abc...
```

You must now ask the system administrator to update the NIS master's `/etc/hosts`, `/etc/bootparams` and `/etc/ethers` files, because of the change in status of the client 'abc.'

```
client abc removed.
```

8.3. System Crashes

Every computer system may crash or hang so that it no longer responds to commands. This section discusses some of the ways you can deal with system hangs and crashes. Topics covered include:

- Your machine's standard procedures after power loss
- How crash dumps are created
- Error messages generated by crash
- How to save the system crash dump is saved
- How to get rid of unwanted crash dumps that can accumulate on your system
- How to force a crash dump
- Steps you can take to analyze a crash dump
- What to do if the automatic reboot fails, and when to call for help.

If you are an advanced administrator, you may additionally want to refer to Chapter 21, "Advanced Administration Topics," for information about the crash kernel debugger.

Standard Machine Behavior after a Power Loss

A major cause of system crash is power loss, due to weather-related problems and power failures.

The `/etc/update` program provides insurance that all will not be lost due to power failure. It is executed by `/etc/rc` during booting; thereafter, `/etc/update` runs the `sync` command every 30 seconds. `sync` updates the superblock and automatically forces the writing of changed blocks to the local disk. This insures that the file system is up-to-date in the event the system crashes.

If the operating system has not attempted to write to the file system for about 30 seconds before losing power, probably nothing abnormal will happen when you try to re-power the system. However, be aware that losing power can always cause problems. If you lose power before the system completes all outstanding disk writes or halt your machine without using `/etc/halt`, `/etc/shutdown`, or `sync`, you may experience problems when your system reboots and runs `fsck`.

The `/usr/etc/fsck` program checks the consistency of your machine's file systems. By default, `fsck` always runs as part of the booting process, unless you reboot using the `fastboot` command. You should always check file system consistency with `fsck` when rebooting. It will attempt to make corrective changes wherever it detects problems. You should let it do so, by answering "yes" to the prompts. If you do not allow `fsck` to make its attempted changes, your file system will be in an unreliable state when you finally boot up.

During its operation, `fsck` may place some unreferenced files in the `lost+found` directory. For more information about how this works and how to retrieve files from `lost+found`, refer to Chapter 24, "File System Check Program," and the `fsck(8)` man page.

System Crash Dumps

When a Sun computer crashes, it attempts to run `sync`. Then it prints out a short message telling why it crashed, attempts to preserve a core image of memory, and invokes an automatic reboot procedure. If no unexpected inconsistency in the file systems is found during reboot, the machine resumes multiuser operations.

Note: Crashing *programs* create *core dumps*, which show the contents of registers when the program crashed.

When your computer panics or crashes, it produces a file known as a *crash dump*, which contains a core image of memory at the time of the crash.

Typically, a bad program should never crash your machine. Sometimes, however, a user program may crash and “hang” something in the system — a user process, a user’s window, or even the whole system — even though it continues to run. Some aspects of a hung system and how to overcome them are discussed below. As the subsection “User Program Crashes” explains below, you are sometimes forced to reboot after a crashed program, even though the operating system has not crashed.

Crash Error Messages

When a workstation crashes, it prints out a message such as:

```
panic:  error message
```

where *error message* is one of the panic error messages described in the `crash(8s)` man page. Less frequently you might see the message

```
Watchdog reset!
```

in place of `panic`. If your machine crashes repeatedly, you should keep exact notes of each message, including punctuation and upper or lowercase lettering. This will help you in repairing the problem.

The file `/var/adm/messages` automatically stores system messages after a crash. If you cannot get a copy of a crash message from the console, look in this file for the information. It may also be helpful in determining patterns of behavior over a period of time. System error messages are documented in *Debugging Tools*.

Enabling Crash Dumps

You can enable crash dumps by editing the `/etc/rc.local` file on your machine. The lines that tell the system to do the crash dump are commented out with pound signs for distribution. To enable crash dumps, locate the following lines in `/etc/rc.local` and remove the pound signs in front of them.

```
mkdir /var/crash/`hostname`
        echo -n 'checking for crash dump... '
intr savecore /var/crash/`hostname`
        echo ''
```

Then change `/var/crash/`hostname`` to the directory where you want your crash dumps placed. Make sure that the partition that `/var` is on is big enough so that the crash dumps can be saved. A crash dump will take up about as much disk space as you have on-board memory.

When the system crashes it writes, or attempts to write, an image of memory into the primary paging partition on disk. After the system is rebooted, the program `savecore` runs and preserves a copy of this core image in the directory `/var/crash/hostname` (or `/var/crash`).

In most cases the system reboots automatically after a crash, cleaning up any problems and allowing you to continue working as before. In cases like this, you probably will not want to save the crash dump.

If you allow crash dumps to accumulate in `/var/crash/hostname` (or `/var/crash`), they will eventually fill up the file system. There is a convenient way to prevent this problem. The `savecore` program reads from a file called `minfree`, in `/var/crash/hostname` (or `/var/crash`) that contains a single number (in ASCII). If the file system contains fewer free kilobytes than the number in `minfree`, then the crash dump will not be saved. If you do use `minfree` to remove the crash dumps, remember to lower its value, or remove the file entirely during those times when you want to save the core image for debugging purposes. `minfree` prevents crash dumps from being saved and never removes them.

Sometimes your machine will hang without crashing. You can usually force a crash dump in this situation. First, abort to the PROM monitor by typing `[L1-a]`, as described in Chapter 5, "Booting Up and Shutting Down Your System." Then type:

```
> g 0
```

The dump will be saved in the `/var/crash/hostname` or `/var/crash` directory according to the `savecore` procedure explained above. However, you cannot force a crash dump in this manner unless you have enabled crash dumps in `rc.local`.

Sometimes the system can get so badly hung that a crash dump cannot be forced. This is especially true on diskless machines because the network has to be operating smoothly to be able to perform the dump.

Analyzing System Crash Dumps

The crash kernel debugger enables you to analyze crash dumps. It is completely described in Chapter 22. The instructions are for advanced administrators and kernel programmers. If you are a novice administrator, do not try to debug the system from the crash dump. However, there may be people at your site who can analyze the crash dumps for you.

User Program Crashes

User program crashes, as distinguished from operating system crashes, are almost always to a bug in the program. The most common messages from a program crash are:

Segmentation Fault

Bus Error

Arithmetic Exception

These often indicate an illegal pointer in the program, perhaps the result of using a value where a pointer to a value was expected.

A crashing program will usually crash dump in its current directory if it has write permission. If it does, you will see the message

Core Dumped

Sometimes the system appears hung or dead; that is, it does not respond to anything you type. Before assuming your program has crashed, check the items below to make sure there is not a simple reason why the system is not responding.

1. Type **CTRL-Q** in case you accidentally pressed (**CTRL-S**). **CTRL-S** freezes the screen.)
2. The `tty` mode may be fouled up. Try typing the line feed character, **CTRL-J**, instead of the **RETURN** key, to force a line feed. If the system responds, type **CTRL-J /usr/ucb/reset CTRL-J** to reset the `tty` modes.
3. If you are running the window system, make sure the mouse cursor resides in the window where you are trying to type commands.
4. Type **CTRL-** (CTRL backslash). This should force a “quit” in the running program, and probably the writing of a core file.
5. Type **CTRL-C** to interrupt the program that may be running.
6. If possible, try logging in to the same CPU from another terminal, or remote log in from another system on the network. Type **ps -ax**, and look for the hung process. If you can identify it, try to kill it. You will have to be superuser or be logged in as the same user running the process. Type **kill <pid number>**. If that does not work try **kill -9 <pid>**. (A quick way to see if a `kill` has worked is to repeat it. If the response is `no such process`, it was killed.)
7. If all of the above fail, abort and reboot.
8. If even that fails, call Customer Service for help.

Crashes Associated with Shared Libraries

The *Programming Utilities and Libraries* manual describes shared libraries fully. This section discusses problems that programmers might have with the runtime linker `ld.so` and shared libraries. If `ld.so` or a widely used shared library, such as `libc.so`, is deleted, most system utilities will fail to execute correctly and the system will become unusable.

As system administrator, you probably will have to install shared libraries and clear any problems associated with them. You must be root to install and delete `ld.so`, as well as any libraries in `/usr/lib`.

Error Messages Associated with Shared Libraries

The following error messages appear if you have problems related to the runtime linker. Recovery procedures follow in the next subsection.

```
** crt0: no /usr/lib/ld.so
** ld.so: lib%s.%d.%d not found
```

These are fatal error messages. They indicate that `ld.so` or a widely-used shared library is missing.

```
** ld.so: open error errno for %s
** ld.so: can't read struct exec for %s
** ld.so: %s is not for this machine type
```

These are fatal error messages, indicating that the shared library executable has been corrupted, was installed with the wrong access rights, or was built to execute on another architecture. The problems indicated by these messages are very rare.

```
** ld.so: warning: %s has older version than expected %d
warning for older minor version used
```

This is a warning message. It occurs when an executable that was previously linked with a newer version of a shared library gets linked against an older version of the library.

Recovering from Problems with Shared Libraries

The following is a partial list of utilities that have been rebuilt without shared libraries so that a system can be restored:

```
ln mount mv rcp restore tar umount
```

To recover from the problems listed above, you need to install a copy of `ld.so` into `/usr/lib`. Note that you must be root to install and delete `ld.so`, as well as any libraries in `/usr/lib`. If you can obtain a copy of `ld.so` but do not have write privileges to `/usr/lib` as root, you can install `ld.so` in another location, such as `/usr/tmp`, then use `mv` to move the linker to `/usr/lib`. Then reboot in single user mode, and use one of the above utilities to restore `ld.so` directly into `/usr/lib`. Then continue booting in multi-user mode.

Installing a New Shared Library

If you have to install a new shared library, it must be assigned a major and minor number. Refer to the *Programming Utilities and Libraries* manual for more information. After a new library is installed it is highly recommended that you run `ldconfig` so that the cache of shared library information is updated.

When To Call Sun for Help

Before calling for help, make sure you have accurately copied down crash messages from the console, or taken them from the `/var/adm` files mentioned above.

If the automatic reboot fails with a message such as:

```
reboot failed: help
```

try to run `fsck` in single user mode. If reboot efforts still fail, call for Tech Support help.

If you are having frequent crashes, gather all the information you can about them, and have it ready when you call for help.

8.4. Making Local Modifications

Locally written commands are typically kept in `/usr/src/local` and their binaries in `/usr/local`. This allows `/usr/bin`, `/usr/ucb`, and `/bin` to correspond to the distribution tape (and to the system manuals). People wishing to use `/usr/local` commands should be made aware that they aren't in the base manual.

A `/usr/junk` directory to throw garbage into, as well as binary directories `/usr/old` and `/usr/new` are useful. The `man` command supports manual directories such as `/usr/man/manj` for junk `/usr/man/manl` for local manual page entries and `/usr/man/mann` for new manual page entries to make this or something similar practical.

8.5. Setting Up and Maintaining `crontab`

Often, you need to have your machine execute certain tasks at a specified time, or to execute routine tasks periodically. A convenient way to do this is to give commands to the clock daemon `/var/etc/cron` to be executed at a specified time. This is done by adding lines to root's `crontab` file which is found in the directory `/var/spool/cron/crontabs`. To modify this file, or create a new file for some user, you must use the `/usr/bin/crontab` editor.

Using the crontab Editor

The crontab editor is invoked with one of the following

```
/bin/crontab [filename]
/bin/crontab -e [username]
/bin/crontab -l [username]
/bin/crontab -r [username]
```

The first syntax above will copy the specified file (or if no file is given, then the standard input) to the `/var/spool/cron/crontabs` directory so that `cron` will run the specified files at the given times.

The second syntax given above will enter the `vi` editor (or if an alternate editor is specified using the `$EDITOR` shell variable, then that will be the editor used), with the resulting file being used as the resultant `crontab` file.

The `crontab` command with the `-l` option lists the `crontab` file for the user. If `username` is given as the argument, then the file for that user will be listed instead.

The `-r` option to `crontab` removes the `crontab` file for the user. If this command is run by the superuser, then any `crontab` file, specified by the `username` argument, can be removed.

Here is a sample file for the login `root`:

Figure 8-1 A Sample crontab File

```
0 * * * * /usr/lib/acct/ckpacct
0 1 * * 1-6 /usr/lib/acct/dodisk
0 2 * * 1-6 /usr/lib/acct/runacct 2> /usr/adm/acct/nite/fd2log
15 5 1 * * /usr/lib/acct/monacct
7 2 * * * /usr/etc/fpa/fparel >/dev/null 2>&1
15 3 * * * find / -name .nfs -mtime +7 -exec rm -f {} ; -o -fstype nfs -prune
5 4 * * 6 /usr/lib/newsyslog >/dev/null 2>&1
15 4 * * * find /var/preserve/ -mtime +7 -a -exec rm -f {} ;
```

Each line in a `crontab` file consists of six fields, separated by spaces or tabs, as follows:

1. Minutes field, which can have values in the range 0 through 59.
2. Hours field, which can have values in the range 0 through 23.
3. Day of the month, in the range 1 through 31.
4. Month of the year, in the range 1 through 12.
5. Day of the week, in the range 0 through 6. Sunday is day 0 in this scheme of things. For backward compatibility with older systems, Sunday may also be specified as day 7.
6. The remainder of the line is the command to be run, including any arguments to that command. A percent character in this field (unless escaped by `\`) is translated to a new-line character. Only the first line (up to a `%` or end

of line) of the command field is executed by the Shell. The other lines are made available to the command as standard input.

Any of fields 1 through 5 can be a list of values separated by commas. A value can either be a number, or a pair of numbers separated by a hyphen, indicating that the job is to be done for all the times in the specified range. If a field is an asterisk character (*) it means that the job is done for all possible values of the field.

Note that the specification of days may be made by two fields (day of the month and day of the week). If both are specified as a list of elements, both are adhered to. For example, `0 0 1,15 * 1` would run a command on the first and fifteenth of each month, as well as on every Monday. To specify days by only one field, the other field should be set to * (for example, `0 0 * * 1` would run a command only on Mondays).

Therefore, for the `crontab` file give above, the third line has the script `/usr/lib/acct/runacct` being run at 2 o'clock in the morning every day except for Sunday, with all messages sent to `stderr` being redirected to the file `/usr/adm/acct/nite/fd2log`. This script processes the main accounting logs so that all fees, connect times, and the like are recorded. For more information on accounting, see the following section "Accounting."

The `cron` utility can be utilized to perform functions every hour, day, week, or even monthly. For further information on using `cron`, see `cron(8)` in the man pages.

8.6. System Log Configuration

The `syslogd` (error logging daemon) is used by many system facilities to record messages whenever an unusual event occurs. Typically these messages are written to `/var/adm/messages`, or your system console, but it is possible to send other messages to specific facilities elsewhere. Messages generated by various programs are redirected to a file. For example, you may wish some messages to be sent to any and all of a selected group of users, or to generally broadcast to everyone who is logged in.

To set up system logging, you must have a file present in the `/etc` directory called `syslog.conf`. This file consists of two columns, the first of which is the error conditions and the second of which is the location or locations where the errors will be logged. The following is a sample `syslog.conf` file which is found in the `/etc` directory when the SunOS is installed:

Figure 8-2 A sample `/etc/syslog.conf` file

```
#
# syslog configuration file.
#
# This file is processed by m4 so be careful to quote (``) names
# that match m4 reserved words. Also, within ifdef's, arguments
# containing commas must be quoted.
#
```

```

# Note: Have to exclude user from most lines so that user.alert
# and user.emerg are not included, because old sendmails
# will generate them for debugging information.  If you
# have no 4.2BSD based systems doing network logging, you
# can remove all the special cases for "user" logging.
#
*.err;kern.debug;auth.notice;user.none      /dev/console
*.err;kern.debug;daemon,auth.notice;mail.crit;user.none /var/adm/messages
lpr.debug                                     /var/adm/lpd-errs

*.alert;kern.err;daemon.err;user.none      operator
*.alert;user.none                            root

*.emerg;user.none                            *

# for loghost machines, to have authentication messages (su, login, etc.)
# logged to a file, un-comment out the following line and adjust the file name
# as appropriate.
#
# if a non-loghost machine chooses to have such messages
# sent to the loghost machine, un-comment out the following line.
#
#auth.notice                                ifdef(`LOGHOST', /var/log/authlog, @loghost)

mail.debug                                  ifdef(`LOGHOST', /var/log/syslog, @loghost)

# following line for compatibility with old sendmails.  they will send
# messages with no facility code, which will be turned into "user" messages
# by the local syslog daemon.  only the "loghost" machine needs the following
# line, to cause these old sendmail log messages to be logged in the
# mail syslog file.
#
ifdef(`LOGHOST',
user.alert                                  /var/log/syslog
)
#
# non-loghost machines will use the following lines to cause "user"
# log messages to be logged locally.
#
ifdef(`LOGHOST', ,
user.err                                    /dev/console
user.err                                    /var/adm/messages
user.alert                                  `root, operator'
user.emerg                                  *
)

```

Notice that in the above `/etc/syslog.conf` file, there are lines that have a `#` in the first column. This indicates that the line is a comment, and is completely ignored by the `syslogd` program.

Possible Error Message Sources

The messages logged by `syslogd` come from a wide variety of events. The term “error message” is a slight misnomer here, because not all of the messages are, in fact, error indicators. Many messages are just routine recordings of the logins, dialups, or any of the other numerous day-to-day activities of an active machine. The message sources are specified with a two part label. The first part is the *originator* of the message, while the second part is the *priority* of the message. A few of the possible originators and priorities are:

syslog.conf originators and priorities			
<i>originator</i>	meaning	<i>priority</i>	meaning
kern	The Kernel	err	All error output
auth	Authentication	debug	Debugging output
daemon	All daemons	notice	Routine output
mail	The mail system	crit	Critical errors
lpr	The Spooling system	emerg	System emergencies
user	User processes	none	don't log output

Therefore, in the figure sample `syslog.conf` above, the following sources are being logged:

sample syslog.conf error sources	
<i>entry</i>	<i>source</i>
*.err	All errors
kern.debug	Kernel debugging output
auth.notice	Root login attempts
daemon.notice	All messages from a daemon
mail.crit	Critical messages from mail
lpr.debug	lpr debugging output
*.alert	All alert messages (shutdowns, etc.)
*.emerg	All emergency messages (panics, etc.)
mail.debug	Mail debugging output
user.none	NO messages pertaining to user processes

In the messages section, an asterisk (*) is a wild card, which applies to all messages of a given type, while a comma can be used to separate various specific parts of a message source identifier, as a short of shorthand. In the above example, the source “daemon,auth.notice” indicates that both the “daemon.notice” and the “auth.notice” messages are to be logged.

The messages that are logged by `syslogd` are logged at the locations specified the second column, which can be a device, or a file, or a group of one or more users. In the instances that an asterisk appears, the message is sent to all applicable recipients.

On a weekly basis, `/var/adm/messages` will be aged by a command in the / file system's crontab file. `/var/adm/messages` will be renamed `/var/adm/messages.0`; `/var/adm/messages.1` will be renamed `/var/adm/messages.2` ; and `/var/adm/messages.2` will be

renamed `/var/adm/messages.3`. The current `/var/adm/messages.3` file, if any, will be deleted. Very old messages are not kept around indefinitely. A new `/var/adm/messages` file will be created, and subsequent error messages will be logged there.

Pre-4.0 Program Logging to syslog Daemon

Pre-4.0 programs will log messages with no facility code but with priorities in the range 1 - 9. Since the current `syslog` accepts priorities in the range 0 - 7, priorities 8 (`LOG_INFO`) and 9 (`LOG_DEBUG`) will look like priorities 0 (`LOG_EMERG`) and 1 (`LOG_ALERT`) from facility 1 (`LOG_USER`). Unfortunately, this will have the effect of making these low priority messages seem to be much higher priority than they really are.

Also, almost all of the values for logging levels have changed. This will cause, for instance, messages logged at the old `LOG_CRIT` level to be logged at the new `LOG_NOTICE` level. In general, old log messages will appear to be less important than intended.

The 4.x `syslog` daemon will force messages that claim to be from the `LOG_KERNEL` facility to look like they came from the `LOG_USER` facility, unless they come from the local kernel. The `syslog.conf` file on the "loghost" machine will be set up to log all `LOG_USER` messages in the log file used to log `LOG_MAIL` messages.

4.1 Program Logging to Pre-4.0 syslog Daemon

All 4.1 programs using `syslog` will send their log messages to the local `syslog` daemon. The default 4.3BSD `syslog` configuration file causes all `syslog` messages to be logged in local files, although it does provide a facility to forward the `syslog` message onto a `syslog` daemon on another machine. This forwarding will be enabled for `sendmail` log messages and perhaps for "authorization system" log messages, forwarding them to "loghost." These forwarded messages will include a facility code in their log message. Except for `LOG_USER | LOG_EMERG` (`== 8`) and `LOG_USER | LOG_ALERT` (`== 9`) (which, by default, will not be forwarded to the `syslog` daemon at "loghost"), these will all cause the priority field in the message to be two digits. The old `syslog` daemon does not understand multi-digit priority fields, and so will log the message with a default priority of `LOG_ERR` (`== 4`). Using the default configuration file, this will cause the message to be logged with all the `sendmail` log messages in `/var/spool/log/syslog`. This is, of course, not nearly so useful as logging to a 4.x `syslog` daemon that would put the message into an appropriate file, but is consistent with the pre-4.0 `syslog`.

Various system daemons and programs record information in the system log to help you to analyze problems. They send this information to the `syslog` daemon on the local machine. The daemon receives these messages and records the information, notifies users of problems, or forwards the information to another machine, typically "loghost." See `syslog(8)` for more details on this process.

Note that dialups do not get logged by `syslog` by default. If this is desired it is necessary to change the `auth.notice` line in the file `/etc/syslog.conf` to read `auth.info`. When a new `syslogd` is started, the change will take effect.

The default configuration runs a `syslog` daemon on each machine, and also keeps all messages on the local machine. If you are running standalone, the default is for the `syslog` daemon to keep all messages on your machine. However, in a network environment it's much easier to track problems if all machines log their information in a single place. Therefore, during first time SunOS installation, the `SunInstall` program reconfigures the logging system so that only the designated server is the `loghost`: `SunInstall` strips the `loghost` alias from the `/etc/hosts` entries for the clients, and adds the alias for the server machine. This means that, for example, if the machine named "unity" is your network server, the beginning of your machines' `/etc/hosts` files might look like:

```
192.9.1.1  unity loghost
192.9.1.2  dynamic
192.9.1.3  string
192.9.1.4  relative
192.9.1.5  dimension
```

Now all messages sent to `loghost` (from the `hosts` `dynamic`, `string`, etc.) are sent to `unity`. There might also be other aliases on the same line of the `/etc/hosts` entry, like `lprhost` or `mailhost`.

If you want to change this configuration — for example, if you have more than one server, and you want only one `loghost` — simply change the placement of the `loghost` alias, and then re-copy `/etc/hosts` to all machines. Test your system log configuration by running:

```
% tail -f /var/spool/log/syslog
```

on the `loghost` machine, then sending any kind of mail on the various other machines. Each message sent will generate four or five lines of output if things are working.

8.7. Accounting

SunOS provides a suite of programs that collect data on system usage by CPU usage, by user, and by process number. The programs include tools for keeping track of connect sessions and disk usage. The accounting utilities can be used for

- charging for system usage
- troubleshooting
- fine tuning performance
- locating and correcting performance problems
- maintaining system security

This chapter deals with how to set up and maintain the accounting utilities, and how to generate summaries of accounting information in the form of reports on the various aspects being monitored.

What Accounting Is

The SunOS accounting package is really a group of programs that record system usage and then provide full reports of that data. The following is an outline the accounting process:

1. While the machine is running, information regarding system use such as login, processes run, and data storage, is collected in various accounting files.
2. Periodically (usually once a day), `/usr/etc/cron` initiates the program `runacct`, which processes the various accounting files and produces a summary file as well as accounting reports. These reports are printed using the `prdaily` program.
3. The cumulative summary files can be processed and printed with the `monacct` program. This program produces a monthly (or other longer term period) report which can be used to track long term usage of a machine.

Accounting Programs

All the C language programs and shell scripts necessary to run the accounting system are in the `/usr/lib/acct` directory. The `acctcom` program is stored in `/bin`. These programs, which are owned by `bin` (except `accton`, which is owned by `root`) do various functions. For example, `/usr/lib/acct/startup` helps initiate the accounting process when the system enters the multi-user mode. To make sure that the `/usr/lib/acct/startup` program runs when it should, make sure that the follow line is not commented out of the `/etc/rc` script

```
#      accounting is off by default.
#
/usr/lib/acct/startup
```

The `chargefee` program is used to charge a particular user for a special service, such as performing a file restore from tape. Other essential programs in the `/usr/lib/acct` directory include `monacct`, `prdaily`, and `runacct`. These and other programs are discussed in more detail in the following sections.

Setting Up Accounting

To run accounting, the system kernel must include the options `sysacct` line. Make sure your kernel has this option if you are trying to make accounting work.

The SunOS operating system records two kinds of accounting information: connect-time accounting and process-resource accounting. Connect-time accounting information is stored in the `/var/adm/wtmp` file. Process-time accounting information is stored in `/var/adm/acct`. These files must either exist or be able to be created for accounting to function.

Also, the `/var/spool/cron/crontabs/root` file must have the following commands added to it:

```
0 * * * * /usr/lib/acct/ckpacct
0 1 * * 1-6 /usr/lib/acct/dodisk
0 2 * * 1-6 /usr/lib/acct/runacct 2> /usr/adm/acct/nite/fd2log
15 5 1 * * /usr/lib/acct/monacct
```

The exact times these commands run may vary, depending on how often you want accounting to be done. See the section of this manual on `cron` for more information on how to set up `crontab` files.

Once these changes have been made and the accounting programs have been installed, accounting will pretty much run on its own.

Daily Accounting

Here is a step-by-step outline of how system accounting works

1. When the system is switched into multi-user mode, the `/usr/lib/acct/startup` program is executed. The startup program executes several other programs that invoke accounting:
 - The `acctwtmp` program adds a “boot” record to `/var/adm/wtmp`. In this record the system name is shown as the login name in the `wtmp` record.
 - The `turnacct` program invokes process accounting. Process accounting begins when the `turnacct` program is executed with the `on` option. The command line `turnacct on` executes the `accton` program with the argument `/var/adm/pacct`.
 - The `remove` shell script “cleans up” the saved `pacct` and `wtmp` files left in the `sum` directory by `runacct`.
2. The `login` and `init` programs record connect sessions by writing records into `/var/adm/wtmp`. Any date changes (using `date` with an argument) are also written to `/var/adm/wtmp`. Reboots and shutdowns (via `acctwtmp`) are also recorded in `/var/adm/wtmp`.
3. When a process ends, the kernel writes one record per process in the `/var/adm/pacct` file in the form of `acct.h`.
4. The two programs that track disk usage, `acctdusg`, and `diskusg`, break down disk usage by login.
5. Every hour, `cron` executes the `ckpacct` program to check the size of `/var/adm/pacct`. If the file grows past 1000 blocks (default), `turnacct switch` is executed. (The `turnacct switch` program moves the `pacct` file and creates a new one.) The advantage of having several smaller `pacct` files becomes apparent when trying to restart `runacct` if a failure occurs when processing these records.
6. If the system is shut down using `shutdown`, the `shutacct` program is executed. The `shutacct` program writes a reason record into `/var/adm/wtmp` and turns off process accounting.
7. If a user requests a service such as a file restore, the `chargefee` program can be used to bill users (specific logins). It adds a record to `/var/adm/fee` that is picked up and processed by the next execution of `runacct` and merged into the total accounting records.
8. `runacct` is executed via `cron` each night. It processes the accounting files — `/var/adm/pacct?`, `/var/adm/wtmp`,

`/var/adm/acct/nite/diskacct`, and `/var/adm/fee` — to produce command summaries and usage summaries by login.

9. The `/usr/lib/acct/prdaily` program is executed on a daily basis by `runacct` to print the daily accounting information collected by `runacct`. The resulting report is stored in `/var/adm/acct/sum/rprt.MMDD`.
10. The `monacct` program should be executed on a monthly basis (or whenever you believe appropriate, such as a fiscal period). The `monacct` program creates a report based on data stored in the `sum` directory that has been updated daily by `runacct`. After creating the report, `monacct` “cleans up” the `sum` directory to prepare the directory’s files for the new `runacct` data.

The `runacct` Program

The main daily accounting shell procedure, `runacct`, is normally invoked by `cron` during non-prime time hours. The `runacct` shell script processes `connect`, `fee`, `disk`, and process accounting files. It also prepares daily and cumulative summary files for use by `prdaily` and `monacct` for billing purposes.

The `runacct` shell script takes care not to damage files if errors occur. A series of protection mechanisms are used that attempt to recognize an error, provide intelligent diagnostics, and end processing in such a way that `runacct` can be restarted with minimal intervention. It records its progress by writing descriptive messages into the file `active`. (Files used by `runacct` are assumed to be in the `/var/adm/acct/nite` directory unless otherwise noted.) All diagnostic output during the execution of `runacct` is written into `fd2log`.

If the files `lock` and `lock1` exist when invoked, `runacct` will complain. These files are used to prevent simultaneous execution of `runacct`. The last-date file contains the month and day `runacct` was last invoked and is used to prevent more than one execution per day. If `runacct` detects an error, a message is written to the console, mail is sent to `root` and `adm`, locks are removed, diagnostic files are saved, and execution is ended.

The `runacct` script takes care not to damage files in the event of errors. A series of protection mechanisms are used that attempt to recognize an error, provide intelligent diagnostics, and terminate processing gracefully. All diagnostic output during the execution of `runacct` is written into `fd2log`.

Re-entrant States of the `runacct` Script

To allow `runacct` to be restartable, processing is broken down into separate reentrant states. A file is used to remember the last state completed. When each state completes, `statefile` is updated to reflect the next state. After processing for the state is complete, `statefile` is read and the next state is processed. When `runacct` reaches the `CLEANUP` state, it removes the locks and ends. States are executed as follows:

- `SETUP`
The command `turnacct switch` is executed to create a new `pacct` file. The process accounting files in `/var/adm/pacct?` (except the `pacct` file) are moved to `/var/adm/Spacct?.MMDD`. The `/var/adm/wtmp` file is moved to

`/var/adm/acct/nite/wtmp.MMDD` with the current time record added on the end and a new `/var/adm/wtmp` is created. `closewtmp` and `utmp2wtmp` add records to `wtmp.MMDD` and the new `wtmp` to account for users currently logged in.

- **WTMPFIX**
The `wtmpfix` program checks the `wtmp.MMDD` file in the `nite` directory for correctness. Because some date changes will cause `acctcon` to fail, `wtmpfix` attempts to adjust the time stamps in the `wtmp` file if a record of a date change appears. It also deletes any corrupted entries from the `wtmp` file. The fixed version of `wtmp.MMDD` is written to `tmpwtmp`.
- **CONNECT**
The `acctcon` program is used to record connect accounting records in the file `ctacct.MMDD`. These records are in `tacct.h` format. In addition, `acctcon` creates the `lineuse` and `reboots` files. The `reboots` file records all the boot records found in the `wtmp` file. `CONNECT` was previously two steps called `CONNECT1` and `CONNECT2`.
- **PROCESS**
The `acctprc` program is used to convert the process accounting files `/var/adm/Spacct?.MMDD`, into total accounting records in `ptacct?.MMDD`. The `Spacct` and `ptacct` files are correlated by number so that if `runacct` fails, the unnecessary reprocessing of `Spacct` files will not occur. One precaution should be noted; when restarting `runacct` in this state, remove the last `ptacct` file because it will not be complete.
- **MERGE**
Merge the process accounting records with the connect accounting records to form `daytacct`.
- **FEES**
Merge in any ASCII `tacct` records from the file `fee` into `daytacct`.
- **DISK**
If the `dodisk` procedure has been run and produced the file `disktacct`, merge the file into `daytacct` and move `disktacct` to `/tmp/disktacct.MMDD`.
- **MERGETACCT**
Merge `daytacct` with `sum/tacct`, the cumulative total accounting file. Each day, `daytacct` is saved in `sum/tacct.MMDD`, so that `sum/tacct` can be recreated if it is corrupted or lost.
- **CMS**
The program `acctcms` is run several times. It is first run to generate the command summary using the `Spacct?` files and writes it to `sum/daycms`. `acctcms` is then run to merge `sum/daycms` with the cumulative command summary file `sum/cms`. Finally, `acctcms` is run to produce the ASCII command summary files `nite/daycms` and `nite/cms` from the files `sum/daycms` and `sum/cms`, respectively.

- **USEREXIT**
Any installation-dependent (local) accounting programs can be included here. `runacct` expects it to be called `/usr/lib/acct/runacct.local`.
- **CLEANUP**
Clean up temporary files, run `prdaily` and save its output in `sum/rprtMMDD`, remove the locks, then exit.

`runacct` Error Messages

The `runacct` procedure can fail for a variety of reasons; the most frequent reasons are a system crash, `/var` running out of space, and a corrupted `wtmp` file. If the `activeMMDD` file exists, check it first for error messages. If the `active` file and lock files exist, check `fd2log` for any mysterious messages. The following are error messages produced by `runacct` and the recommended recovery actions:

- **ERROR: locks found, run aborted**
The files `lock` and `lock1` were found. These files must be removed before `runacct` can restart.
- **ERROR: acctg already run for date: check**
`/var/adm/acct/nite/lastdate`
The date in `lastdate` and today's date are the same. Remove `lastdate`.
- **ERROR: turnacct switch returned rc=?**
Check the integrity of `turnacct` and `accton`. The `accton` program must be owned by `root` and have the `setuid` bit set.
- **ERROR: Spacct?.MMDD already exists. File setups probably already run.**
Check status of files, then run setups manually, if necessary.
- **ERROR: /var/adm/acct/nite/wtmp.MMDD already exists, run setup manually**
`/var/adm/wtmp` has already been copied to
`/var/adm/acct/nite/wtmp.MMDD`
- **ERROR: wtmpfix errors see**
`/var/adm/acct/nite/wtmperror`
`Wtmpfix` detected a corrupted `wtmp` file. Use `fwtmp` to correct the corrupted file.
- **ERROR: invalid state, check**
`/var/adm/acct/nite/statefile`
The file `statefile` is probably corrupted. Check `statefile` and read `active` before restarting.

Files Produced by runacct

The following files produced by runacct (found in `/var/adm/acct`) are of particular interest:

- `nite/lineuse`
runacct calls `acctcon` to gather data on terminal line usage from `/var/adm/acct/nite/tmpwtmp` and writes the data to `/var/adm/acct/nite/lineuse`. `prdaily` uses this data to report line usage. This report is especially useful for detecting bad lines. If the ratio between the number of logoffs to logins exceeds about 3:1, there is a good possibility that the line is failing.
- `nite/daytacct`
This file is the total accounting file for the day in `tacct.h` format.
- `sum/tacct`
This file is the accumulation of each day's `nite/daytacct` and can be used for billing purposes. It is restarted each month or fiscal period by the `monacct` procedure.
- `sum/daycms`
runacct calls `acctcms` to process the data about the commands used during the day. This information is stored in `/var/adm/acct/sum/daycms`. It contains the daily command summary. The ASCII version of this file is `/var/adm/acct/nite/daycms`.
- `sum/cms`
This file is the accumulation of each day's command summaries. It is restarted by the execution of `monacct`. The ASCII version is `nite/cms`.
- `sum/loginlog`
runacct calls `lastlogin` to update the last date logged in for the logins in `/var/adm/acct/sum/loginlog`. `lastlogin` also removes from this file logins that are no longer valid.
- `sum/rprtMMDD`
Each execution of runacct saves a copy of the daily report that was printed by `prdaily`.

Fixing Corrupted Files

Unfortunately, this accounting system is not entirely error free. Occasionally, a file will become corrupted or lost. Some of the files can simply be ignored or restored from the `filesave` backup. However, certain files must be fixed to maintain the integrity of the accounting system.

Fixing wttmp Errors

The `wttmp` files seem to cause the most problems in the day-to-day operation of the accounting system. When the date is changed and the system is in multi-user mode, a set of date change records is written into `/var/adm/wtmp`. The `wtmpfix` program is designed to adjust the time stamps in the `wtmp` records when a date change is encountered. However, some combinations of date changes and reboots will slip through `wtmpfix` and cause `acctcon` to fail. The following steps show how to patch up a `wtmp` file.

Figure 8-3 *Repairing a wtmp File*

```
# cd /var/adm/acct/nite
# fwtmp < wtmp.MMDD > xwtmp
# vi xwtmp
  delete corrupted records or
  delete all records from beginning
  up to the date change
# fwtmp -ic < xwtmp > wtmp.MMDD
```

If the `wtmp` file is beyond repair, create a null `wtmp` file. This will prevent any charging of connect time. As a side effect, the lack of a `wtmp` file prevents `acctprc` from identifying the login that owned a particular process; the process is charged to the owner of the first login in the password file for the appropriate user ID.

Fixing `tacct` Errors

If the installation is using the accounting system to charge users for system resources, the integrity of `sum/tacct` is important. Occasionally, mysterious `tacct` records will appear with negative numbers, duplicate user IDs, or a user ID of 65,535. First, check `sum/tacctprev`, using `prtacct` to print it. If it looks all right, the latest `sum/tacct.MMDD` should be patched up, then `sum/tacct` recreated. A simple patch up procedure would be the following.

Figure 8-4 *Repairing a tacct File*

```
# cd /var/adm/acct/sum
# acctmerg -v < tacct.MMDD > xtacct
# vi xtacct
  remove the bad records
  write duplicate uid records to another file
# acctmerg -i < xtacct > tacct.MMDD
# acctmerg tacctprev < tacct.MMDD > tacct
```

The current `sum/tacct` can be recreated by merging all existing `tacct.MMDD` files by using `acctmerg`, since the `monacct` procedure removes all the old `tacct.MMDD` files.

Restarting `runacct`

Called without arguments, `runacct` assumes that this is the first invocation of the day. The argument `MMDD` is necessary if `runacct` is being restarted and specifies the month and day for which `runacct` will rerun the accounting. The entry point for processing is based on the contents of `statefile`. To override `statefile`, include the desired state on the command line. The following are some sample procedures.

To start `runacct`:

```
# nohup runacct 2> /var/adm/acct/nite/fd2log&
```

To restart `runacct`:

```
# nohup runacct 0601 2> /var/adm/acct/nite/fd2log&
```

To restart `runacct` at a specific state:

```
# nohup runacct 0601 WTMPFIX 2> /var/adm/acct/nite/fd2log&
```

Billing Users

The `chargefee` program stores charges for special services provided to a user, such as file restores, in the file `fee`. This file is incorporated by `runacct` every day.

To register special fees, enter the following command:

```
# chargefee loginID amount
```

where *amount* is an integer amount to be charged. Most locations prefer to set up their own shell scripts for this function, with codes for services rendered. The operator then need identify only the service rendered; the system can tabulate the charge.

The monthly accounting program `monacct` produces monthly summary reports similar to those produced daily. The `monacct` program also summarizes the accounting information into the files in the `/var/adm/acct/fiscal` directory. This information can be used to generate monthly billing.

Setting Up Non-Prime Time Discounts

System accounting provides facilities to give users a discount for non-prime time system use. For this to work, you must inform the accounting system of the dates of holidays and the hours that are considered non-prime time, such as weekends. To do this, you must edit the `/usr/lib/acct/holidays` file that contains the prime/non-prime table for the accounting system. The format is composed of three types of entries:

- **Comment Lines**—Comment lines are marked by an asterisk in the first column of the line. Comment lines may appear anywhere in the file.
- **Year Designation Line**—This line should be the first data line (noncomment line) in the file and must appear only once. The line consists of three fields of four digits each (leading white space is ignored). For example, to specify the year as 1989, prime time start at 9:00 A.M., and non-prime time start at 4:30 P.M., the following entry would be appropriate:

```
1989 0900 1630
```

A special condition allowed for in the time field is that the time 2400 is automatically converted to 0000.

- **Company Holidays Lines**—These entries follow the year designation line and have the following general format:

```
Date      Description of Holiday
```

The date field has the format *month/day* and indicates the date of the holiday. The holiday field is actually commentary and is not currently used by other programs.

Daily Accounting Reports

The `runacct` shell script generates four basic reports upon each invocation. They cover the areas of connect accounting, usage by login on a daily basis, command usage reported by daily and monthly totals, and a report of the last time users were logged in. The four basic reports generated are:

- The daily report shows line utilization by tty number.
- The daily usage report indicates usage of system resources by users.
- The daily command summary indicates usage of system resources by commands, listed in descending order of use of memory (in other words, the command that used the most memory is listed first). This same information is reported for the month with the monthly command summary.
- The last login shows the last time each user logged in (arranged in chronological order).

The following paragraphs describe the reports and the meanings of the data presented in each one.

Daily Report

This report gives information about each terminal line used.

Figure 8-5 *Sample Daily Report*

```
Jun 29 09:53 1989  DAILY REPORT FOR oboef Page 1

from Wed Jun 28 17:45:22 1989
to   Thu Jun 29 09:51:25 1989
1   runacct
1   acctcon

TOTAL DURATION IS 946 MINUTES
LINE      MINUTES  PERCENT  # SESS  # ON # OFF
term/23   25 3    7    4    4
term/22   157 16   6    3    3
TOTALS    183 --  13   7    7
```

The `from/to` lines tell you the time period reflected in the report. The times are the time the last accounting report was generated until the time the current accounting report was generated. It is followed by a log of system reboots, shutdowns, power fail recoveries, and any other record dumped into `/var/adm/wtmp` by the `acctwtmp` program.

The second part of the report is a breakdown of line utilization. The `TOTAL DURATION` tells how long the system was in multi-user state (accessible through the terminal lines). The columns are:

- `LINE`: the terminal line or access port
- `MINUTES`: the total number of minutes that line was in use during the accounting period
- `PERCENT`: the total number of `MINUTES` the line was in use, divided into the `TOTAL DURATION`
- `# SESS`: the number of times this port was accessed for a `login` session
- `# ON`: This column does not have much meaning any more. It used to list the number of times that a port was used to log a user on; but because `login` can no longer be executed explicitly to log in a new user, this column should be identical with `SESS`.
- `# OFF`: This column reflects not just the number of times a user logs off but also any interrupts that occur on that line. Generally, interrupts occur on a port when `ttymon` is first invoked when the system is brought to multi-user state. Where this column does come into play is when the `# OFF` exceeds the `# ON` by a large factor. This usually indicates that the multiplexer, modem, or cable is going bad, or there is a bad connection somewhere. The most common cause of this is an unconnected cable dangling from the multiplexer.

During real time, monitor `/var/adm/wtmp` because it is the file from which the connect accounting is geared. If the `wtmp` file grows rapidly, execute `acctcon -l file < /var/adm/wtmp` to see which `tty` line is the noisiest. If the interrupting is occurring at a furious rate, general system performance will be affected.

Daily Usage Report

The daily usage report gives a breakdown of system resource utilization by user.

Figure 8-6 Sample Daily Usage Report

Jun 29 09:53 1989 DAILY USAGE REPORT FOR oboef Page 1

UID	LOGIN NAME	CPU (MINS)		KCORE-MINS		CONNECT (MINS)		DISK BLOCKS	# OF PROCS	# OF SESS	# DISK SAMPLES	FEE	
		PRIME	NPRIME	PRIME	NPRIME	PRIME	NPRIME						
0	TOTAL	5	12	6	16	131	51	0	1114	13	0	0	
0	root	2	8	1	11	0	0	0	519	0	0	0	
3	sys	0	1	0	1	0	0	0	45	0	0	0	
4	adm	0	2	0	1	0	0	0	213	0	0	0	
5	uucp	0	0	0	0	0	0	0	53	0	0	0	
7987	usef	0	0	0	1	20	14	0	15	6	0	0	

The data provided include the following:

- UID: The user ID
- LOGIN NAME: The login name of the user. This information is useful because it identifies a user who has multiple login IDs.
- CPU (MINS): This represents the amount of time the user's process used the central processing unit. This category is broken down into PRIME and NPRIME (non-prime) utilization. The accounting system's idea of this breakdown is located in the /etc/acct/holidays file.
- KCORE-MINS: This represents a cumulative measure of the amount of memory a process uses while running. The amount shown reflects kilobyte segments of memory used per minute. This measurement is also broken down into PRIME and NPRIME amounts.
- CONNECT and (MINS): This identifies the amount of "real time" used. What this column really identifies is the amount of time that a user was logged into the system. If the amount of time is high and the number shown in the column # OF PROCS is low, you can safely conclude that the owner of the login for which the report is being generated is a "line hog." That is, this person logs in first thing in the morning and hardly touches the terminal the rest of the day. Watch out for this kind of user. This column is also subdivided into PRIME and NPRIME utilization.
- DISK BLOCKS: When the disk accounting programs have been run, the output is merged into the total accounting record (daytacct) and shows up in this column. This disk accounting is accomplished by the program acctdusg. For accounting purposes, a "block" is 512 bytes.
- # OF PROCS: This column reflects the number of processes that were invoked by the user. This is a good column to watch for large numbers indicating that a user may have a shell procedure that has run out of control.
- # OF SESS: The number of times a user logged on to the system is shown in this column.

- # DISK SAMPLES: This indicates how many times the disk accounting was run to obtain the average number of DISK BLOCKS listed earlier.
- FEE: An often unused field in the total accounting record, the FEE field represents the total accumulation of widgets charged against the user by the chargefee shell procedure. The chargefee procedure is used to levy charges against a user for special services performed such as file restores, and so on.

Daily Command Summary

The daily command summary report shows the system resource utilization by command. With this report, you can identify the most heavily used commands and, based on how those commands use system resources, gain insight on how best to tune the system. The daily command and monthly reports are virtually the same except that the daily command summary reports only on the current accounting period while the monthly total command summary tells the story for the start of the fiscal period to the current date. In other words, the monthly report reflects the data accumulated since the last invocation of `monacct`.

These reports are sorted by `TOTAL KCOREMIN`, which is an arbitrary yardstick but often a good one for calculating “drain” on a system.

Figure 8-7 *Sample Daily Command Summary*

Jun 29 09:52 1989 DAILY COMMAND SUMMARY Page 1

TOTAL COMMAND SUMMARY									
COMMAND NAME	NUMBER	TOTAL KCOREMIN	TOTAL CPU-MIN	TOTAL REAL-MIN	MEAN SIZE-K	MEAN CPU-MIN	HOG FACTOR	CHARS TRNSFD	BLOCKS READ
TOTALS	1114	2.44	16.69	136.33	0.15	0.01	0.12	4541666	1926
cs	227	1.01	2.45	54.99	0.41	0.01	0.04	111025	173
oboe	10	0.50	2.06	9.98	0.24	0.21	0.21	182873	223
vi	12	0.35	0.62	44.23	0.55	0.05	0.01	151448	60
sed	143	0.09	0.82	1.48	0.10	0.01	0.55	14505	35
sadc	13	0.08	0.19	1.45	0.44	0.01	0.13	829088	19
more	3	0.04	0.07	2.17	0.59	0.02	0.03	30560	1
cut	14	0.03	0.09	0.28	0.37	0.01	0.33	154	13
uudemon.	76	0.03	0.66	2.30	0.05	0.01	0.29	43661	13
ypmatch	29	0.03	0.30	0.72	0.08	0.01	0.42	80765	35
mail	4	0.02	0.06	0.09	0.37	0.01	0.60	4540	9
ckstr	21	0.02	0.11	0.13	0.17	0.01	0.85	0	4
awk	13	0.02	0.12	0.21	0.15	0.01	0.54	444	2
ps	2	0.02	0.10	0.13	0.17	0.05	0.77	8060	21
find	9	0.02	3.35	5.73	0.00	0.37	0.58	355269	760
ls	1	0.01	0.19	0.24	0.08	0.19	0.80	564224	4
rn	2	0.01	0.01	0.06	1.02	0.01	0.22	0	9
mv	24	0.01	0.14	0.17	0.10	0.01	0.81	3024	36

The data provided include the following:

- COMMAND NAME
The name of the command. Unfortunately, all shell procedures are lumped together under the name `sh` because only object modules are reported by the process accounting system. It's a good idea to monitor the frequency of programs called `a.out` or `core` or any other name that does not seem quite right. Often people like to work on their favorite version of backgammon, but they do not want everyone to know about it. `acctcom` is also a good tool to use for determining who executed a suspiciously named command and also if superuser privileges were used.
- PRIME NUMBER CMDS
The total number of invocations of this particular command during prime time.
- NON-PRIME NUMBER CMDS
The total number of invocations of this particular command during non-prime time.
- TOTAL KCOREMIN
The total cumulative measurement of the amount of kilobyte segments of

memory used by a process per minute of run time.

- PRIME TOTAL CPU-MIN
The total processing time this program has accumulated during prime time.
- NON-PRIME TOTAL CPU-MIN
The total processing time this program has accumulated during non-prime time.
- PRIME TOTAL REAL-MIN
Total real-time (wall-clock) minutes this program has accumulated. This total is the actual “waited for” time as opposed to kicking off a process in the background during prime time.
- NON-PRIME TOTAL REAL-MIN
Total real-time (wall-clock) minutes this program has accumulated. This total is the actual “waited for” time as opposed to kicking off a process in the background during non-prime time.
- MEAN SIZE-K
This is the mean of the TOTAL KCOREMIN over the number of invocations reflected by NUMBER CMDS.
- MEAN CPU-MIN
This is the mean derived between the NUMBER CMDS and TOTAL CPU-MIN.
- HOG FACTOR
The total CPU time divided by the elapsed time. This shows the ratio of system availability to system utilization. This gives a relative measure of the total available CPU time consumed by the process during its execution.
- CHARS TRNSFD
This column, which may go negative because of overflow, is a total count of the number of characters pushed around by the read and write system calls.
- BLOCKS READ
A total count of the physical block reads and writes that a process performed.

Total Command Summary

The monthly command summary is similar to the daily command summary. The only difference is that the monthly command summary shows totals accumulated since the last invocation of `monacct`.

Figure 8-8 Sample Total Command Summary

TOTAL COMMAND SUMMARY									
COMMAND NUMBER	TOTAL	TOTAL	TOTAL	MEAN	MEAN	HOG	CHARS	BLOCKS	
NAME	CMDS	KCOREMIN	CPU-MIN	REAL-MIN	SIZE-K	CPUMIN	FACTOR	TRNSFD	READ
TOTALS	301314	300607.70	4301.59	703979.81	69.88	0.01	0.01	6967631360	10596385
troff	480	58171.37	616.15	1551.26	94.41	1.28	0.40	650669248	194926
pnews	5143	29845.12	312.20	1196.93	95.59	0.06	0.26	1722128384	2375741
uucp	2710	16625.01	212.95	52619.21	78.07	0.08	0.00	228750872	475343
nroff	1613	15463.20	206.54	986.06	74.87	0.13	0.21	377563304	277957
vi	3040	14641.63	157.77	14700.13	92.80	0.05	0.01	116621132	206025
expire	14	13424.81	104.90	265.67	127.98	7.49	0.39	76292096	145456
comp	3483	12140.64	60.22	423.54	201.62	0.02	0.14	9584838	372601
ad_d	71	10179.20	50.02	1158.31	203.52	0.70	0.04	11385054	19489
as	2312	9221.59	44.40	285.52	207.68	0.02	0.16	35988945	221113
gone	474	8723.46	219.93	12099.01	39.67	0.46	0.02	10657346	19397
il0	299	8372.60	44.45	454.21	188.34	0.15	0.10	60169932	78664
find	760	8310.97	196.91	728.39	42.21	0.26	0.27	58966910	710074
ld	2288	8232.84	61.19	425.57	134.55	0.03	0.14	228701168	279530
egrep	832	7585.34	62.62	199.11	121.14	0.08	0.31	22119268	37196
csh	56314	7538.40	337.60	291655.70	22.33	0.01	0.00	93262128	612892
du	624	5049.58	126.32	217.59	39.97	0.20	0.58	16096269	215297
ls	12690	4765.60	75.71	541.53	62.95	0.01	0.14	65759473	207920
rn	52	4235.71	28.11	959.74	150.70	0.54	0.03	28291679	28285
.									
.									
.									

Last Login Report

This report simply gives the date when a particular login was last used. You can use this information to find unused logins and login directories that may be archived and deleted.

Figure 8-9 *Sample Last Login*

Feb 13 04:40 1988 LAST LOGIN Page 1

```

00-00-00  **RJE** 90-01-01  jlr    90-02-09  cec42   90-02-13  cec20
00-00-00  **rje** 90-01-13  crom   90-02-10  jgd     90-02-13  cec22
00-00-00  sunol   90-01-14  usg     90-02-10  wbr     90-02-13  cec23
00-00-00  adm     90-01-17  cec11   90-02-11  cec30   90-02-13  cec24
00-00-00  daemon 90-01-17  cec38   90-02-11  cec41   90-02-13  cec25
00-00-00  notes   90-01-17  cec40   90-02-11  cec43   90-02-13  cec26
00-00-00  ras     90-01-18  cec60   90-02-11  cec53   90-02-13  cec27
00-00-00  ppd     90-01-19  cec35   90-02-11  cec54   90-02-13  cec3
00-00-00  absolme 90-01-19  cec37   90-02-11  cec55   90-02-13  cec31
00-00-00  rfe     90-01-22  dmkm    90-02-11  cec56   90-02-13  cec32
00-00-00  sher    90-01-26  ask     90-02-11  cec57   90-02-13  cec4
00-00-00  sys     90-01-26  cec39   90-02-11  cec58   90-02-13  cec6
00-00-00  trouble 90-01-27  sync    90-02-11  jwg     90-02-13  cec7
00-00-00  ussr    90-02-02  pk1     90-02-11  skt     90-02-13  cec8
00-00-00  uucp    90-02-03  ibm     90-02-11  tfm     90-02-13  commlp
00-00-00  wna     90-02-03  slk     90-02-12  cec21   90-02-13  djs
89-07-06  lp      90-02-04  cec59   90-02-12  cec28   90-02-13  epic
89-07-30  djenner 90-02-05  cec33   90-02-12  cec29   90-02-13  jab
89-08-19  bldg    90-02-05  cec34   90-02-12  csp     90-02-13  jcs
89-12-08  etna    90-02-05  cec36   90-02-12  drc     90-02-13  mak
90-01-14  s       90-02-05  cec51   90-02-12  emw     90-02-13  mdn
90-01-09  rbb     90-02-05  dfh     90-02-12  je      90-02-13  mlp
90-01-25  dmf     90-02-05  fsh     90-02-12  kab     90-02-13  nbh
90-01-25  epda    90-02-05  pkw     90-02-12  rap     90-02-13  rah
.
.
.

```

Looking at the `pacct` File with `acctcom`

At any given time, the contents of the `/var/adm/pacct?` files or any file with records in the `acct.h` format may be examined using the `acctcom` program. If you don't specify any files and don't provide any standard input when you run this command, `acctcom` reads the `pacct` file. Each record read by `acctcom` represents information about a dead process (active processes may be examined by running the `ps` command). The default output of `acctcom` provides the following information: the name of the command (prepended with a # if string executed with superuser privileges), the user, tty name (listed as ? if unknown), starting time, ending time, real time (in seconds), cpu (in seconds), and mean size (in K). The following information can be obtained by using options: F (the fork/exec flag: 1 for fork without exec), STAT (the system exit status), HOG FACTOR, KCORE MIN, CPU FACTOR, CHARS TRNSFD, and BLOCKS READ.

The options are:

- Show some average statistics about the processes selected (printed after the output records)

- Read the file(s) backward, showing latest commands first. (Has no effect if reading standard input.)
- Print the fork/exec flag and system exit status columns.
- Instead of mean memory size, show the hog factor, which is the fraction of total available CPU time consumed by the process during its execution. Hog factor = (total CPU time)/(elapsed time).
- Print columns containing the number of physical I/O operations in the output.
- Show total kcore-minutes instead of memory size.
- Show mean core size (shown by default unless superseded by another option).
- Show CPU factor (user time)/(system-time + user-time)
- Show separate system and user CPU times.
- Exclude column headings from the output.
- Show only processes belonging to the terminal `/dev/line`
- Show only processes belonging to user.
- Show only processes belonging to group.
- Show processes existing at or after time, given in the format `hr[:min[:sec]]`.
- Show processes existing at or before time, given in the format `hr[:min[:sec]]`.
- Show processes starting at or after time, given in the format `hr[:min[:sec]]`.
- Show processes starting at or before time, given in the format `hr[:min[:sec]]`. Using the same time for both `-S` and `-E` shows processes that existed at the time.
- Show only commands matching pattern (a regular expression as in `ed` except that “+” means one or more occurrences).
- Don't print output records, just print averages (akin to `-a`).
- Instead of printing the records, copy them in `acct.h` format to `ofile`.
- Show only processes that exceed *factor*, where *factor* is the “hog factor” explained in the description of the `-h` option.
- Show only processes with CPU system time exceeding *sec* seconds.
- Show only processes with total CPU time, system plus user, exceeding *sec* seconds.
- Show only processes transferring more characters than the cut-off number specified by *chars*.

Accounting Files

The `/var/adm` directory structure contains the active data collection files and is owned by the `adm` login (currently user ID of 4).

A brief description of the files found in the `/var/adm` directory follows:

- `dtmp`
output from the `acctdusg` program
- `fee`
output from the `chargefee` program, ASCII `tacct` records
- `pacct`
active process accounting file
- `pacct?`
process accounting files switched via `turnacct`
- `Spacct?.MMDD`
process accounting files for `MMDD` during execution of `runacct`

The `/var/adm/acct` directory contains the `nite`, `sum`, and `fiscal` directories, which contain the actual data collection files. For example, the `nite` directory contains files that are reused daily by the `runacct` procedure. A brief summary of the files in the `/var/adm/acct/nite` directory follows:

- `active`
Used by `runacct` to record progress and print warning and error messages. `activeMMDD` same as `active` after `runacct` detects an error
- `cms`
ASCII total command summary used by `prdaily`
- `ctacct.MMDD`
Connect accounting records in `tacct.h` format
- `ctmp`
Output of `acctcon1` program, connect session records in `ctmp.h` format.
- `daycms`
ASCII daily command summary used by `prdaily`
- `daytacct`
Total accounting records for one day in `tacct.h` format
- `disktacct`
disk accounting records in `tacct.h` format, created by the `dodisk` procedure
- `fd2log`
diagnostic output during execution of `runacct`; see “Setting Up Accounting” at the beginning of this chapter
- `lastdate`
last day `runacct` executed (in `date +%m%d` format)
- `lock lockl`
used to control serial use of `runacct`

- `lineuse`
tty line usage report used by `prdaily`
- `log`
diagnostic output from `acctcon`
- `logMMDD`
same as `log` after `runacct` detects an error
- `reboots`
contains beginning and ending dates from `wtmp` and a listing of reboots
- `statefile`
used to record current state during execution of `runacct`
- `tmpwtmp`
`wtmp` file corrected by `wtmpfix`
- `wtmperror`
place for `wtmpfix` error messages
- `wtmperrorMMDD`
same as `wtmperror` after `runacct` detects an error
- `wtmp.MMDD`
`runacct`'s copy of the `wtmp` file

The `sum` directory contains the cumulative summary files updated by `runacct` and used by `monacct`. A brief summary of the files in the `/var/adm/acct/sum` directory follows:

- `cms`
total command summary file for current fiscal in internal summary format
- `cmsprev`
command summary file without latest update
- `daycms`
command summary file for the day's usage in internal summary format
- `loginlog`
record of last date each user logged on; created by `lastlogin` and used in the `prdaily` program
- `rprtMMDD`
saved output of `prdaily` program
- `tacct`
cumulative total accounting file for current fiscal
- `tacctprev`
same as `tacct` without latest update
- `tacct.MMDD`
total accounting file for `MMDD`

The `fiscal` directory contains periodic summary files created by `monacct`. A brief description of the files in the `/var/adm/acct/fiscal` directory follows:

- `cms?`
total command summary file for fiscal ? in internal summary format
- `fiscrpt?`
report similar to `rprt?` for fiscal ?
- `tacct?`
total accounting file for fiscal ?

Reconfiguring the System Kernel

It is a good idea to reconfigure your system's kernel after installing Release 4.1. Since the generic kernel that SunOS Release 4.1 provides supports more devices than previous releases, it is significantly larger and occupies considerably more memory. You can significantly improve system performance by tailoring the generic kernel or using the small, machine-specific kernels to reconfigure your kernel.

Small kernels are designed to fit the hardware and software of your system to provide maximum performance. All kernels for each architecture contain descriptions to help you decide the kernel entries your system needs. Reconfiguring the kernel is not a difficult task. Carefully following the instructions in the section *Kernel Reconfiguration Procedures*, and in the README file, (located in the `/usr/kvm/sys/sun[3,3x,4,4c]/conf` directory) will give you a working kernel tailored to your system.

This chapter discusses the following subjects:

- Reasons to reconfigure the system's kernel.
- Sample sections of a `GENERIC` kernel configuration file from a broad perspective.
- Procedures to place the reconfigured kernel into operation.
- Procedures to change swap space.

9.1. Why Reconfigure the Kernel?

There are two reasons to reconfigure the kernel:

- To free up memory that would otherwise be used by unused kernel modules, thus improving your system's performance.
- To tell the kernel about the hardware you added after installation or the software packages that require kernel modification to support.

The SunOS Release 4.1 tapes contain the kernel configuration file for your Sun workstation. When you build the kernel from the kernel file supplied on the release tape, the utilities involved create code that supports all hardware and software devices available for your Sun workstation architecture. Since the kernel is unnecessarily large; your particular system probably does not need all those items. Also, on machines with a small amount of memory, using the kernel configuration file on the release tape wastes large amounts of main memory,

which may seriously degrade system performance.

Modifying a copy of the `GENERIC` configuration file or using a small kernel provides a kernel modified to those items that apply only to your configuration. This smaller, customized kernel takes up less space in memory, giving larger effective memory size to programs. This improves system performance substantially, particularly if you need to run SunView and other large applications or programs.

You also reconfigure the kernel when you make major hardware or software additions to your system. For example, you edit the kernel configuration file when you add new hardware, such as a second disk or graphics controller. In addition, you need to edit the kernel configuration file when you add major software packages that you did not select when you initially ran SunInstall. For example, if you decide to attach your standalone configuration to the network file system (NFS), you have to enable this option from the configuration file.

9.2. Parts of a Kernel Configuration File

This section examines a kernel configuration file from both a broad and narrow perspective. First, it explains how the configuration file is used during the kernel building process. Then an overview is given that describes the major sections of a configuration file.

The Reconfiguration Process

Building a new system is a semi-automatic process. Most of it is handled by a configuration-build utility called `/usr/etc/config`, which generates the files you need to compile and link the kernel. Your major activity in the configuration process is to create the kernel configuration file `SYSTEM_NAME`. `SYSTEM_NAME` is usually the name of your machine or other identifying name. This file contains a description of the kernel you want `/usr/etc/config` to produce. `/usr/etc/config` then uses this information to create the directory `/usr/kvm/sys/sun[3,3x,4,4c]/SYSTEM_NAME` and builds the kernel there.

Rather than creating the configuration file from scratch, you can copy and edit the `GENERIC` configuration file or use one of the small kernels your release provides in `/usr/kvm/sys/sun[3,3x,4,4c]/conf`. `GENERIC` is a configuration file that contains all possible entries of devices that Sun supplies for your model of Sun workstation. You may need to edit this file extensively to fit you needs. You can also select one of the small kernels that require little if any editing.

Major Sections of the `GENERIC` Configuration File

The `GENERIC` configuration file is divided into three sections:

- A system identification section that identifies the type of machine you have, including the name that you want to give the kernel. This section is similar for the `GENERIC` configuration file for each model type.
- A connections section that tells the kernel which CPU board and bus connections are available for attaching controllers.
- A devices section that contains lines specific to each type of controller, disk, tape, or other type device board that you want to configure.

To understand the format a configuration file, it is best to begin by examining `GENERIC`. Note that the pound sign character (`#`) starts a comment that continues

to the end of the line. Here is an example of the first few lines of the Sun-3 `GENERIC` file.

```
#
# @(#) GENERIC from master 1.8 90/01/11 SMI
#
# This config file describes a generic Sun-3 kernel, including all
# possible standard devices and software options.
#
# The following lines include support for all Sun-3 CPU types.
# There is little to be gained by removing support for particular
# CPUs, so you might as well leave them all in.
#
machine      "sun3"
cpu         "SUN3_50"   # Sun-3/50
cpu         "SUN3_60"   # Sun-3/60
cpu         "SUN3_110"  # Sun-3/110
cpu         "SUN3_160"  # Sun-3/75, Sun-3/140, Sun-3/160
cpu         "SUN3_260"  # Sun-3/260, Sun-3/280
cpu         "SUN3_E"    # Sun-3E (Eurocard VMEbus cpu)

#
# Name this kernel GENERIC.
#
ident       "GENERIC"

#
# This kernel supports about 8 users. Count one user for each
# timesharing user, one for each window that you typically use, and one
# for each diskless client you serve. This is only an approximation used
# to control the size of various kernel data structures, not a hard limit.
#
maxusers    8

#
# Include all possible software options.
#
# The INET option is not really optional; every kernel must include it.
#
options     INET      # basic networking support - mandatory

#
# The following options are all filesystem related. You only need
# QUOTA if you have UFS. You only need UFS if you have a disk.
# Diskless machines can remove QUOTA, UFS, and NFSERVER. LOFS and TFS
# are only needed if you're using the Sun Network Software Environment.
# HSFS is only needed if you have a CD-ROM drive and want to access
# ISO-9660 or High Sierra format CD discs.
#
options     QUOTA     # disk quotas for local disks
```

```
options UFS      # filesystem code for local disks
options NFSCLIENT # NFS client side code
options NFSSERVER # NFS server side code
options LOFS     # loopback filesystem - needed by NSE
options TFS     # translucent filesystem - needed by NSE
options TMPFS   # tmp (anonymous memory) file system
options HSFS    # High Sierra (ISO 9660) CD-ROM file system

#
# The following options are for accounting and auditing.  SYSAUDIT
# should be removed unless you are using the C2 security features.
#
options SYSACCT      # process accounting, see acct(2) & sa(8)
options SYSAUDIT    # C2 auditing for security

#
# The following options are for various System V IPC facilities.
# Most standard software does not need them, although they are
# used by SUNGKS and some third-party software.
#
options IPCMESSAGE # System V IPC message facility
options IPCSEMAPHORE # System V IPC semaphore facility
options IPCSHMEM  # System V IPC shared-memory facility

#
# The following option is only needed if you want to use the trpt
# command to debug TCP problems.
#
options TCPDEBUG # TCP debugging, see trpt(8)

#
# The following option includes the software DES support, needed if
# you're using secure NFS or secure RPC and you don't have a DES chip.
#
options CRYPT     # software encryption

#
# The following two options are only needed if you want to use RFS.
#
options RFS
options VFSSTATS

#
# The following two options are needed for asynchronous I/O.
#
options LWP      # kernel threads
options ASYNCHIO # asynch I/O (requires LWP)

#
# The following option adds support for loadable kernel modules.
#
options VDDRV    # loadable modules
```

```

#
# The following option adds support for the old SCSI architecture.
#
options OLDSCSI      # Old SCSI architecture - mandatory

#
# The following option adds support for SunView 1 journaling.
#
options WINSVJ      # SunView 1 journaling support

#
# Build one kernel based on this basic configuration.
# It will use the generic swap code so that you can have
# your root filesystem and swap space on any supported device.
# Put the kernel configured this way in a file named "vmunix".
#
config vmunix swap generic

#
# Include support for all possible pseudo-devices.
#
# The first few are mostly concerned with networking.
# You should probably always leave these in.
#

```

You can see by these line groupings that the configuration file has three different types of entries:

- Lines that give a general description of the system (parameters global to the kernel image that this configuration generates)
- A line that describes items specific to each kernel image generated
- Lines that describe the devices on the system and connections to which these devices are attached.

The System Identification Section

The system identification section, which is shown in the illustration above, contains general system description lines and system-specific lines. They are described below.

General System Description Lines

The first six general description lines in the configuration file are mandatory for every Sun workstation. They are:

machine type

This field tells the kernel that the system is to run on the machine type specified. The legal *types* for Sun workstations are "sun3", "sun3x", "sun4", and "sun4c". Note that the double quotes are considered part of the description.

cpu type

This field indicates that the system is to run on the CPU type specified. More than one CPU type can appear in the configuration file.

CPU types for a Sun-3 machine are:

```

machine      "sun3"
cpu          "SUN3_50"      # Sun-3/50
cpu          "SUN3_60"      # Sun-3/60
cpu          "SUN3_110"     # Sun-3/110
cpu          "SUN3_160"     # Sun-3/75, Sun-3/140, Sun-3/160
cpu          "SUN3_260"     # Sun-3/260, Sun-3/280
cpu          "SUN3_E"       # Sun-3E (Eurocard VMEbus cpu)

```

This field tells the kernel the name you wish for the system identifier—the name for the machine or machines that run this kernel. You must include *name* in double quotes if it contains any numbers (for example, "SDST120"), or you will get a syntax error when you run `/usr/etc/config`. If the name is `GENERIC`, the kernel name will be taken from the configuration file name.

Note that when editing the `GENERIC` file, you do not have to name your kernel `GENERIC`. If you specify `options generic`, then you must use the `swap generic` clause on the `config` line. When you specify these two, the kernel prompts you during the boot process for the location of `/root` and `/swap`, for example, on the local disk or from an NFS server.

maxusers number

This field tells the kernel that the maximum expected number of simultaneously active users on this system is *number*. The number you specify is used to size several system data structures. The recommended value for `maxusers` on a server, regardless of model type is 8. The `GENERIC` and small configuration files for each model type give further instructions to help determine a specification for `maxusers` that is appropriate to your system's needs. Beware that entering a number that is too large will bloat your kernel, so consider carefully what your needs are.

options type

The fields beginning with the word `options` tell the kernel to compile the selected software options into the system. *type* has a different value for each options line. For example, here are several options lines in the Sun-3 `GENERIC` kernel:

```

#
options QUOTA      # disk quotas for local disks
options UFS        # filesystem code for local disks
options NFSCLIENT  # NFS client side code
options NFSSERVER  # NFS server side code
options LOFS       # loopback filesystem - needed by NSE
options TFS        # translucent filesystem - needed by NSE
options TMPFS      # tmp (anonymous memory) file system
options HSFS       # High Sierra (ISO 9660) CD-ROM file system
#

```

Refer to the `GENERIC` configuration file for your system to determine which options are appropriate for it.

System-Specific Description Lines

The next type of line in the kernel configuration file is a single line specifying the name of the file the kernel build procedure will create.

The line has the following syntax:

```
config kernelname config_clauses
```

Here *kernelname* indicates the name of the loaded kernel image. Its value is usually *vmunix*.

config_clauses are one or more specifications indicating where the root file system is located and where the primary paging (or swap) device is located. A *config_clause* may be one or more of the following:

```
root [on] root_device
```

This clause specifies the location of the root file system.

```
swap [on] swap_device
```

This clause specifies the location of the primary swapping and paging area. Specifying `swap generic` enables you to place your root file system and swap space on any supported device. When specifying options `generic`, you must specify `swap generic`, as well.

```
dumps [on] dump_device
```

This clause specifies where the `/export/dump` kernel should place core images after a crash.

The “on” in the syntax of each clause is optional. Separate multiple *config_clauses* by white space. For example, the *config* line for a system with root on its first SMD disk (Partition a) and swap on Partition b of the same disk might be:

```
config vmunix root on xy0a swap on xy0b
```

Note also that the device names supplied in the clauses may be fully specified—as a device, unit, and file system partition—or underspecified. If underspecified, the `config` program uses built-in rules to select default unit numbers and file system partitions. (The *Advanced Administration Topics* chapter explains rules that are followed for underspecified location of devices.) For example, the swap partition need not be specified at all if the root device is specified. This is because the default is to place the `/swap` partition in Partition b of the same disk where the root file system is located. Thus you could use the following *config_clause* to represent the same information as the previous clause:

```
config vmunix root on xy0
```

For diskless clients:

Use the following `config_clause`

```
config vmunix root on type nfs
```

The Pseudo-Devices Section

This section lists all possible *pseudo devices* for your model. A pseudo-device is a collection of programs or a device driver that has no associated hardware. For example, here are three pseudo-devices needed to run SunView 1

```
#
# The next few are for SunWindows support, needed to run SunView 1.
#
pseudo-device  win128          # window devices, allow 128 windows
pseudo-device  dtop4           # desktops (screens), allow 4
pseudo-device  ms              # mouse support
#
```

The `GENERIC` configuration file gives suggestions as to which pseudo-devices you may need.

The Connections Section

The next section in the configuration file lists the possible on board and bus connections, grouped together by the machine model. These connections, in conjunction with controllers, devices, and disks form a structure that enables your system to recognize various hardware attached to it. For each device or controller on a bus, you need to select the bus type it is connected to, as listed under connections for your machine type.

Sun-3, Sun-3x, and Sun-4 have separate connection lists for each model, or group of models, as you will see if you examine their `GENERIC` configuration files. Here is a segment from the Sun-3 `GENERIC` kernel connections section.

```
# connections for machine type 1 (SUN3_160)
controller    virtual 1 at nexus ? # virtually addressed devices
controller    obmem 1 at nexus ?   # memory-like devices on the cpu board
controller    obio 1 at nexus ?    # I/O devices on the cpu board
controller    vme16d16 1 at nexus ? # VME 16 bit address 16 bit data devices
controller    vme24d16 1 at nexus ? # VME 24 bit address 16 bit data devices
controller    vme32d16 1 at nexus ? # VME 32 bit address 16 bit data devices
controller    vme16d32 1 at nexus ? # VME 16 bit address 32 bit data devices
controller    vme24d32 1 at nexus ? # VME 24 bit address 32 bit data devices
controller    vme32d32 1 at nexus ? # VME 32 bit address 32 bit data devices
```

```
# connections for machine type 2 (SUN3_50)
controller      virtual 2 at nexus ?
controller      obmem 2 at nexus ?
controller      obio 2 at nexus ?
```

Note that machine type 1 is a Sun-3/160 and machine type 2 is a Sun-3/50. The first three connections, `virtual`, `obmem`, and `obio` are used by Sun for specific devices. For example, the `fpa` floating point accelerator uses the `virtual` connection, and various graphics controllers use `obmem` and `obio`. For machine type 1, connections prefaced with “`vme`” are on the VMEbus. For example, the phrase `vme32d16` indicates a 32 bit VMEbus with 16 bit data. Note however that the Sun-3/50 does not have a VME connection listed.

The easiest way to modify the connections section is to leave as is all connections lines listed for your machine type. Then, comment out each connection line for all other machine types. That way, as you add controllers and devices, the connections are already enabled and will be recognized by your system.

The Devices Section

The final section of the configuration file lists all devices that can be supported by a Sun-3, Sun-3x, or Sun-4. Devices are grouped into controllers and, if applicable, the disks and tapes that may be connected to them. Each device is listed on a separate device description line. The basic format of these lines is described as follows.

Device Description Lines

When reconfiguring the kernel configuration file, you need to specify each device on your machine so that the generated kernel recognizes these devices during the boot process. Devices may be hardware-related entities, that is, controller boards and devices attached to the controllers, or software pseudo-devices.

The device description lines tell the system what devices to look for and use, and how these devices are inter-connected. Each line has the following syntax:

```
dev_type dev_name at connect_dev more info
```

Below is a definition of each parameter on the device description line.

dev_type

This item specifies the device type. *dev_type* may be one of the following:

- **controller.** Usually a disk or tape controller.
- **disk or tape.** Devices connected to a controller.
- **device.** Hardware entity attached to the main system bus, for example, an Ethernet controller board.
- **pseudo-device.** Software subsystems or drivers with no associated hardware, such as the `pseudo-tty` driver and various network subsystems, such as the NFS and Internet subsystems.

<i>dev_name</i>	Standard device name and unit number of the device you are specifying (if the device is not a pseudo-device). For example, <i>dev_name</i> for the first Xylogics disk controller on a system is <code>xyc0</code> .
<i>connect_dev</i>	Connection to which this device is attached. Here are the possible connections for all Sun workstation configurations:
<code>virtual</code>	Virtual preset
<code>obmem</code>	On-board memory
<code>obio</code>	On-board I/O
<code>vme16d16</code> (<code>vme16</code>)	VMEbus: 16 bit address/16 bit data (Sun-3 Sun-3x and Sun-4)
<code>vme24d16</code> (<code>vme24</code>)	VMEbus: 24 bit address/16 bit data (Sun-3 Sun-3x and Sun-4)
<code>vme32d16</code>	VMEbus: 32 bit address/16 bit data (Sun-3 and Sun-4)
<code>vme16d32</code>	VMEbus: 16 bit address/32 bit data (Sun-3 Sun-3x and Sun-4)
<code>vme24d32</code>	VMEbus: 24 bit address/32 bit data (Sun-3 Sun-3x and Sun-4)
<code>vme32d32</code>	VMEbus: 32 bit address/32 bit data (Sun-3 Sun-3x and Sun-4)

When modifying the configuration file, you must specify the connections that apply to your system, but do not need to specify connections that don't apply. If you are unsure of the type of system bus or data bus you have, refer to the hardware manuals that came with the system.

more_info This is a sequence of the following:

```
csr addr drive number flags number priority level vector intr number
```

The above arguments are completely described in *Advanced Administration Topics*, because you need to supply values for them only if you are going to configure your own device drivers.

Briefly *csr addr* specifies the address of the csr (command and status registers) for a device. *drive number* specifies which drive the line applies to. *flags number*, *priority level*, and *vector intr number* are all values defined in the device driver.

Here is a sample set of lines describing Xylogics 450/451 disk controllers and disks.

```

controller xyc0 at vme16d16 ? csr 0xee40 priority 2 vector xyintr 0x48
controller xyc1 at vme16d16 ? csr 0xee48 priority 2 vector xyintr 0x49
disk xy0 at xyc0 drive 0
disk xy1 at xyc0 drive 1
disk xy2 at xyc1 drive 0
disk xy3 at xyc1 drive 1

```

Bus types and devices must hang off the appropriate controller, which in turn hangs off another controller until a configuration is formed that gets you to a bus type that hangs off a “nexus.” Notice how each line in the connections section concludes with the words “at nexus.” On Sun systems, all bus types are considered to hang off a nexus. For example, the following SMD disk :

```
disk xy0 at xyc0 drive 0
```

is attached to the Xylogics controller:

```
controller xyc0 at vme16d16 ? csr 0xee40 priority 2 vector xyintr 0x48
```

that is attached to the bus type vme16d16, as listed in the following entry from the connections section:

```
controller vme16d16 1 at nexus ?
```

In order to determine and note which standard devices are present on your machine, boot the GENERIC kernel after executing SunInstall. If you want, you can delete, or comment out, those lines that pertain to devices not on your machine. Or you can configure your file with the devices that are on other machines that you will possibly want to boot from the same kernel.

9.3. Modifying the Kernel Configuration Files

Note: Some parameters relating to the System V Inter-Process Communication (IPC) extensions may also be tuned in the configuration file. These parameters do not appear in the GENERIC file but are documented in *Advanced Administration Topics*.

Modification Procedures

This subsection shows the annotated GENERIC kernel configuration files for each model of Sun computer. Note how the GENERIC file contains comments that suggest which entries to pick for your particular system.

If the comments indicate that the line is **mandatory**, you *must* include it in every system configuration file, either exactly as it stands, or, if commentary indicates variables, with the variables adjusted to fit your system. Some options shown as mandatory are only required if you have other related options selected for your system.

Here are suggested procedures for modifying the GENERIC kernel configuration file.

Step 1.

Go through the next subsections and find the copy of GENERIC that pertains to your model of Sun computer.

Step 2.

Read the annotated `GENERIC` file and determine how you want to modify each line. You can modify a line by doing one of the following:

- Changing its parameters so that it applies to your configuration. Usually you do this for lines indicated as **mandatory**. For example, the `maxusers` line is mandatory, but you can change its value from the default.
- “Commenting out” a line if it does not apply to your current configuration, but may apply to it in the future. To do this, type a pound sign (`#`) at the beginning of the line. The utilities that build the kernel ignore lines beginning with the pound sign.

For example, suppose you have a SCSI-2 controller with one disk and one tape drive. You might want to comment out the lines that apply to a second SCSI-2 controller, second disk, and second tape drive. At a later date, you may add some or all of this equipment. All you need to do to have this equipment recognized is to remove the pound sign, then make the new kernel.

If you need to determine what controllers you have, enter:

```
% /etc/dmesg
```

Something like the following appears:

```
Oct 19 14:55
SunOS Release 4.1 (PATCH_SDST80) #7: Wed Aug 23 16:26:37 PDT 1989
Copyright (c) 1989 by Sun Microsystems, Inc.
mem = 8192K (0x800000)
avail mem = 7430144
Ethernet address = 8:0:20:7:37:32
sm0 at obio 0x66000000 pri 2
st0 at sm0 slave 32
sd0 at sm0 slave 0
sd0: <Toshiba MK 156F cyl 815 alt 2 hd 10 sec 34>
sd2 at sm0 slave 8
sd2: <Quantum ProDrive 105S cyl 974 alt 2 hd 6 sec 35>
sd6 at sm0 slave 24
sd6: <Quantum ProDrive 105S cyl 974 alt 2 hd 6 sec 35>
le0 at obio 0x65002000 pri 3
zs0 at obio 0x62002000 pri 3
zs1 at obio 0x62000000 pri 3
bwtwo0 at obmem 0x50300000 pri 4
bwtwo0: resolution 1152 x 900
pp0 at obio 0x6f000000 pri 1
WARNING: TOD clock not initialized -- CHECK AND RESET THE DATE!
root on sd6a fstype 4.2
swap on sd6b fstype spec size 15540K
dump on sd6b fstype spec size 15512K
```

The above message shows everything loaded in the kernel whether it is attached or not if it is a SCSI device. All other disks show what is actually attached with a device description following the device:

```
sd0: <Toshiba MK 156F cyl 815 alt 2 hd 10 sec 34>
```

The above message shows a Sun-3/80 with two internal disk drives and one attached SCSI disk.

- Deleting any lines that will never apply to your configuration. For example, if you have a diskless client, you might want to delete lines applying to Xylogics disk and tape controllers. These controllers are often used with servers. If you do add a disk or tape to your machine, it will probably be enclosed in a “shoebox” containing SCSI controller, disk, and possibly, tape drive.

Step 3.

If you wish, mark up each line in the text, indicating the changes you want to make to the actual file.

Step 4.

Type the following:

```
# cd /usr/sys/sun[3,3x,4,4c]/conf
```

to go to the directory containing the GENERIC kernel configuration file.

Step 5.

Copy the file `GENERIC`. Call the new file `SYS_NAME`, where `SYS_NAME` represents the name you want to give to your system. Use the following command:

```
# cp GENERIC SYS_NAME
```

If your customized kernel is already in use and you now want to modify it, you should copy the customized kernel configuration file and edit the copy.

Step 6.

Change the permissions for `SYS_NAME` as follows:

```
# chmod +w SYS_NAME
```

Step 7.

Edit `SYS_NAME` using your preferred text editor. Use the notes you made as a guide while you make these changes. Make sure to include the proper device description lines for your machine.

Now you are ready to begin the reconfiguration process. Go on to the next major section.

The Sun-3 GENERIC Kernel

The following example is the entire GENERIC configuration file for a Sun-3.

```
#
# @(#) GENERIC from master 1.8 90/01/11 SMI
#
# This config file describes a generic Sun-3 kernel, including all
# possible standard devices and software options.
#
# The following lines include support for all Sun-3 CPU types.
# There is little to be gained by removing support for particular
# CPUs, so you might as well leave them all in.
#
machine      "sun3"
cpu          "SUN3_50"   # Sun-3/50
cpu          "SUN3_60"   # Sun-3/60
cpu          "SUN3_110"  # Sun-3/110
cpu          "SUN3_160"  # Sun-3/75, Sun-3/140, Sun-3/160
cpu          "SUN3_260"  # Sun-3/260, Sun-3/280
cpu          "SUN3_E"    # Sun-3E (Eurocard VMEbus cpu)

#
# Name this kernel GENERIC.
#
ident        "GENERIC"

#
# This kernel supports about 8 users.  Count one user for each
# timesharing user, one for each window that you typically use, and one
# for each diskless client you serve.  This is only an approximation used
# to control the size of various kernel data structures, not a hard limit.
#
maxusers     8

#
# Include all possible software options.
#
# The INET option is not really optional; every kernel must include it.
#
options INET      # basic networking support - mandatory

#
# The following options are all filesystem related.  You only need
# QUOTA if you have UFS.  You only need UFS if you have a disk.
# Diskless machines can remove QUOTA, UFS, and NFSERVER.  LOFS and TFS
# are only needed if you're using the Sun Network Software Environment.
# HSFS is only needed if you have a CD-ROM drive and want to access
# ISO-9660 or High Sierra format CD discs.
#
options QUOTA     # disk quotas for local disks
options UFS       # filesystem code for local disks
options NFSCLIENT # NFS client side code
options NFSERVER  # NFS server side code
```



```

options LOFS      # loopback filesystem - needed by NSE
options TFS      # translucent filesystem - needed by NSE
options TMPFS    # tmp (anonymous memory) file system
options HSFS     # High Sierra (ISO 9660) CD-ROM file system

#
# The following options are for accounting and auditing.  SYSAUDIT
# should be removed unless you are using the C2 security features.
#
options SYSACCT  # process accounting, see acct(2) & sa(8)
options SYSAUDIT # C2 auditing for security

#
# The following options are for various System V IPC facilities.
# Most standard software does not need them, although they are
# used by SUNGKS and some third-party software.
#
options IPCMESSAGE # System V IPC message facility
options IPCSEMAPHORE # System V IPC semaphore facility
options IPCSHMEM # System V IPC shared-memory facility

#
# The following option is only needed if you want to use the trpt
# command to debug TCP problems.
#
options TCPDEBUG # TCP debugging, see trpt(8)

#
# The following option includes the software DES support, needed if
# you're using secure NFS or secure RPC and you don't have a DES chip.
#
options CRYPT # software encryption

#
# The following two options are only needed if you want to use RFS.
#
options RFS
options VFSSTATS

#
# The following two options are needed for asynchronous I/O.
#
options LWP # kernel threads
options ASYNCHIO # asynch I/O (requires LWP)

#
# The following option adds support for loadable kernel modules.
#
options VDDRV # loadable modules

#
# The following option adds support for the old SCSI architecture.
#

```

```
options OLDSCSI      # Old SCSI architecture - mandatory

#
# The following option adds support for SunView 1 journaling.
#
options WINSVJ       # SunView 1 journaling support

#
# Build one kernel based on this basic configuration.
# It will use the generic swap code so that you can have
# your root filesystem and swap space on any supported device.
# Put the kernel configured this way in a file named "vmunix".
#
config vmunix swap generic

#
# Include support for all possible pseudo-devices.
#
# The first few are mostly concerned with networking.
# You should probably always leave these in.
#
pseudo-device  pty      # pseudo-tty's, also needed for SunView
pseudo-device  ether    # basic Ethernet support
pseudo-device  loop     # loopback network - mandatory

#
# The next few are for SunWindows support, needed to run SunView 1.
#
pseudo-device  win128   # window devices, allow 128 windows
pseudo-device  dtop4    # desktops (screens), allow 4
pseudo-device  ms       # mouse support

#
# The following is needed to support the Sun keyboard, with or
# without the window system.
#
pseudo-device  kb       # keyboard support

#
# The following is needed to support the Sun dialbox.
#
pseudo-device  db       # dialbox support

#
# The following is for asynchronous tty support for the ALM-2 (aka MCP).
# If you have an ALM-2 (MCP) and it is being used to connect timesharing
# terminals, you will need this.
# The number appended to mcpa should be the total number of serial lines
# provided by the ALM-2s in the system.  For example, if you have four
# ALM-2s this should read "mcpa64".
#
pseudo-device  mcpa128
```

```

#
# The following is for the streams pipe device, which is required by RFS.
#
pseudo-device  sp      # streams pipe device

#
# The following are for streams NIT support.  NIT is used by
# etherfind, traffic, rarpd, and ndbootd.  As a rule of thumb,
# NIT is almost always needed on a server and almost never
# needed on a diskless client.
#
pseudo-device  snit      # streams NIT
pseudo-device  pf        # packet filter
pseudo-device  nbuf      # NIT buffering module

#
# The following is for the "clone" device, used with streams devices.
# This is required if you include streams NIT support or RFS.
#
pseudo-device  clone     # clone device

#
# The following is for TLI.  Include these if you want to use RFS and/or
# the TLI library.
#
pseudo-device  tim64
pseudo-device  tirw64

#
# The following is for the TCP TLI stream head.  This provides a TLI-conforming
# interface on top of which you can run RFS and/or the TLI library.  This is
# required by RFS.
#
pseudo-device  tcptli32

#
# The following sections describe what kinds of buses each cpu type supports.
# You should never need to change this.  (The word "nexus" is historical.)
#
# Controller (bus) descriptions:
#
# virtual    virtually addressed devices
# obmem      memory-like devices on the cpu board
# obio       I/O devices on the cpu board
# vme16d16   VME 16 bit address 16 bit data devices
# vme24d16   VME 24 bit address 16 bit data devices
# vme32d16   VME 32 bit address 16 bit data devices
# vme16d32   VME 16 bit address 32 bit data devices
# vme24d32   VME 24 bit address 32 bit data devices
# vme32d32   VME 32 bit address 32 bit data devices
#
# connections for machine type 1 (SUN3_160)

```

```
controller virtual 1 at nexus ?
controller obmem 1 at nexus ?
controller obio 1 at nexus ?
controller vme16d16 1 at nexus ?
controller vme24d16 1 at nexus ?
controller vme32d16 1 at nexus ?
controller vme16d32 1 at nexus ?
controller vme24d32 1 at nexus ?
controller vme32d32 1 at nexus ?

# connections for machine type 2 (SUN3_50)
controller virtual 2 at nexus ?
controller obmem 2 at nexus ?
controller obio 2 at nexus ?

# connections for machine type 3 (SUN3_260)
controller virtual 3 at nexus ?
controller obmem 3 at nexus ?
controller obio 3 at nexus ?
controller vme16d16 3 at nexus ?
controller vme24d16 3 at nexus ?
controller vme32d16 3 at nexus ?
controller vme16d32 3 at nexus ?
controller vme24d32 3 at nexus ?
controller vme32d32 3 at nexus ?

# connections for machine type 4 (SUN3_110)
controller virtual 4 at nexus ?
controller obmem 4 at nexus ?
controller obio 4 at nexus ?
controller vme16d16 4 at nexus ?
controller vme24d16 4 at nexus ?
controller vme32d16 4 at nexus ?
controller vme16d32 4 at nexus ?
controller vme24d32 4 at nexus ?
controller vme32d32 4 at nexus ?

# connections for machine type 7 (SUN3_60)
controller virtual 7 at nexus ?
controller obmem 7 at nexus ?
controller obio 7 at nexus ?

# connections for machine type 8 (SUN3_E)
controller virtual 8 at nexus ?
controller obmem 8 at nexus ?
controller obio 8 at nexus ?
controller vme16d16 8 at nexus ?
controller vme24d16 8 at nexus ?
controller vme32d16 8 at nexus ?
controller vme16d32 8 at nexus ?
controller vme24d32 8 at nexus ?
controller vme32d32 8 at nexus ?
```

```

#
# The following (large) section describes the standard devices supported
# by this kernel.
#
#
# Support for 4 Xylogics 7053 controllers with 4 drives each.
#
controller  xdc0 at vme16d32 ? csr 0xee80 priority 2 vector xdintr 0x44
controller  xdc1 at vme16d32 ? csr 0xee90 priority 2 vector xdintr 0x45
controller  xdc2 at vme16d32 ? csr 0xeea0 priority 2 vector xdintr 0x46
controller  xdc3 at vme16d32 ? csr 0xeeb0 priority 2 vector xdintr 0x47
disk        xd0 at xdc0 drive 0
disk        xd1 at xdc0 drive 1
disk        xd2 at xdc0 drive 2
disk        xd3 at xdc0 drive 3
disk        xd4 at xdc1 drive 0
disk        xd5 at xdc1 drive 1
disk        xd6 at xdc1 drive 2
disk        xd7 at xdc1 drive 3
disk        xd8 at xdc2 drive 0
disk        xd9 at xdc2 drive 1
disk        xd10 at xdc2 drive 2
disk        xd11 at xdc2 drive 3
disk        xd12 at xdc3 drive 0
disk        xd13 at xdc3 drive 1
disk        xd14 at xdc3 drive 2
disk        xd15 at xdc3 drive 3
#
# Support for 2 Xylogics 450/451 controllers with 2 drives each.
#
controller  xyc0 at vme16d16 ? csr 0xee40 priority 2 vector xyintr 0x48
controller  xyc1 at vme16d16 ? csr 0xee48 priority 2 vector xyintr 0x49
disk        xy0 at xyc0 drive 0
disk        xy1 at xyc0 drive 1
disk        xy2 at xyc1 drive 0
disk        xy3 at xyc1 drive 1
#
# Support for the SCSI-2 host adapter with 2 disks and 1 1/4" tape
# on the first SCSI controller, 2 disks and 1 1/4" tape on the second
# SCSI controller, 2 embedded SCSI disks, and a CD-ROM drive.
#
controller  sc0 at vme24d16 ? csr 0x200000 priority 2 vector scintr 0x40
tape        st0 at sc0 drive 040 flags 1
tape        st1 at sc0 drive 050 flags 1
disk        sr0 at sc0 drive 060 flags 2
disk        sd0 at sc0 drive 000 flags 0
disk        sd1 at sc0 drive 001 flags 0
disk        sd2 at sc0 drive 010 flags 0
disk        sd3 at sc0 drive 011 flags 0
disk        sd4 at sc0 drive 020 flags 0

```

```

disk      sd6 at sc0 drive 030 flags 0

#
# Support for the scsi-E host adapter used with the Sun-3/E.
#
controller se0 at vme24d16 ? csr 0x300000 priority 2 vector se_intr 0x40
tape       st0 at se0 drive 040 flags 1
tape       st1 at se0 drive 050 flags 1
disk       sr0 at se0 drive 060 flags 2
disk       sd0 at se0 drive 000 flags 0
disk       sd1 at se0 drive 001 flags 0
disk       sd2 at se0 drive 010 flags 0
disk       sd3 at se0 drive 011 flags 0
disk       sd4 at se0 drive 020 flags 0
disk       sd6 at se0 drive 030 flags 0

#
# Support for the scsi-3 host adapter and the on-board scsi controller
# on several machines (e.g. 3/50).
#
controller si0 at vme24d16 ? csr 0x200000 priority 2 vector siintr 0x40
controller si0 at obio ? csr 0x140000 priority 2
controller si1 at vme24d16 ? csr 0x204000 priority 2 vector siintr 0x41
tape       st0 at si0 drive 040 flags 1
tape       st1 at si0 drive 050 flags 1
tape       st2 at si1 drive 040 flags 1
tape       st3 at si1 drive 050 flags 1
disk       sr0 at si0 drive 060 flags 2
disk       sd0 at si0 drive 000 flags 0
disk       sd1 at si0 drive 001 flags 0
disk       sd2 at si0 drive 010 flags 0
disk       sd3 at si0 drive 011 flags 0
disk       sd4 at si0 drive 020 flags 0
disk       sd6 at si0 drive 030 flags 0

#
# Support for the 2 tty lines (ttya, ttyb) on the cpu board.
# Needed when using a terminal for the console device.
# Flags=3 says to supply carrier in software for both lines.
# Change flags by changing /etc/ttytab; see ttysoftcar(8).
#
device     zs0 at obio ? csr 0x20000 flags 3 priority 3

#
# Support for the keyboard and mouse interface. Needed when
# using a frame buffer as the console device or with SunView.
# You can remove this line if you don't use the standard Sun
# Workstation keyboard and mouse, but if you leave it in don't
# change it.
#
device     zs1 at obio ? csr 0x00000 flags 0x103 priority 3

#

```

```

# Support for 4 ALMs (Systech MTI-800/1600).  Flags set for
# all lines to be local, i.e., carrier supplied by software
# rather than by the device.
# Change flags by changing /etc/ttytab; see ttysoftcar(8).
#
device      mti0 at vme16d16 ? csr 0x620 flags 0xffff priority 4
           vector mtiintr 0x88
device      mti1 at vme16d16 ? csr 0x640 flags 0xffff priority 4
           vector mtiintr 0x89
device      mti2 at vme16d16 ? csr 0x660 flags 0xffff priority 4
           vector mtiintr 0x8a
device      mti3 at vme16d16 ? csr 0x680 flags 0xffff priority 4
           vector mtiintr 0x8b

#
# Support for 8 MCP boards.
# Note that the first four MCPs use the same vectors as the ALMs and thus
# ALMs cut into the total number of MCPs that can be installed.
# Make sure the maxusers line above is correct for the number of
# users expected.
#
device      mcp0 at vme32d32 ? csr 0x01000000 flags 0x1ffff priority 4
           vector mcpintr 0x8b
device      mcp1 at vme32d32 ? csr 0x01010000 flags 0x1ffff priority 4
           vector mcpintr 0x8a
device      mcp2 at vme32d32 ? csr 0x01020000 flags 0x1ffff priority 4
           vector mcpintr 0x89
device      mcp3 at vme32d32 ? csr 0x01030000 flags 0x1ffff priority 4
           vector mcpintr 0x88
device      mcp4 at vme32d32 ? csr 0x02000000 flags 0x1ffff priority 4
           vector mcpintr 0xa0
device      mcp5 at vme32d32 ? csr 0x02010000 flags 0x1ffff priority 4
           vector mcpintr 0xa1
device      mcp6 at vme32d32 ? csr 0x02020000 flags 0x1ffff priority 4
           vector mcpintr 0xa2
device      mcp7 at vme32d32 ? csr 0x02030000 flags 0x1ffff priority 4
           vector mcpintr 0xa3

#
# Support for the on-board Intel 82586 Ethernet chip on many machines.
#
device      ie0 at obio ? csr 0xc0000 priority 3

#
# Support for the Sun-3/E Intel Ethernet board for Sun-3/E cpu systems.
#
device      ie0 at vme24d16 8 csr 0x31ff02 priority 3 vector ieintr 0x74

#
# Support for a second Intel Ethernet interface, using a Multibus
# Ethernet board with a Multibus-to-VME adapter.
#
device      ie1 at vme24d16 ? csr 0xe88000 priority 3 vector ieintr 0x75

```

```

# Support for additional "Sun-3/E SCSI/Ethernet" boards
#device      ie2 at vme24d16 ? csr 0x31ff02 priority 3 vector ieintr 0x76
#device      ie3 at vme24d16 ? csr 0x35ff02 priority 3 vector ieintr 0x77

#
# Support for the on-board LANCE Ethernet chip (Am7990) on many machines.
#
device       le0 at obio ? csr 0x120000 priority 3

#
# Support for 2 Ciprico TapeMaster tape controllers with 1 tape drive each.
#
controller   tm0 at vme16d16 ? csr 0xa0 priority 3 vector tmintr 0x60
controller   tm1 at vme16d16 ? csr 0xa2 priority 3 vector tmintr 0x61
tape         mt0 at tm0 drive 0 flags 1
tape         mt1 at tm1 drive 0 flags 1

#
# Support for 2 Xylogics 472 tape controllers with 1 tape drive each.
#
controller   xtc0 at vme16d16 ? csr 0xee60 priority 3 vector xtintr 0x64
controller   xtc1 at vme16d16 ? csr 0xee68 priority 3 vector xtintr 0x65
tape         xt0 at xtc0 drive 0 flags 1
tape         xt1 at xtc1 drive 0 flags 1

#
# Support for color frame buffers on various machine types.
#
# 3/110 on-board 8-bit frame buffer
device       cgfour0 at obmem 4 csr 0xff000000 priority 4 # 3/110

# 3/60 P4 8-bit color frame buffer
device       cgfour0 at obmem 7 csr 0xff300000 priority 4 # 3/60

# 3/60 plug-in 8-bit color frame buffer
device       cgfour0 at obmem 7 csr 0xff400000 priority 4 # 3/60

# 3/E 8-bit color frame buffer
device       cgfour0 at obmem 8 csr 0xff400000 priority 4 # 3/E

# 3/60 P4 accelerated 8-bit color frame buffer
device       cgsix0 at obmem 7 csr 0xff000000 priority 4 # 3/60

# 3/60 P4 24-bit color frame buffer
device       cgeight0 at obmem 7 csr 0xff300000 priority 4 # 3/60

#
# Support for monochrome frame buffers on various machine types.
#
device       bwtwo0 at obmem 1 csr 0xff000000 priority 4 # 3/160
device       bwtwo0 at obmem 2 csr 0x100000 priority 4 # 3/50
device       bwtwo0 at obmem 3 csr 0xff000000 priority 4 # 3/260

```



```

# 3/110 on-board frame buffer overlay plane
device      bwtwo0 at obmem 4 csr 0xff000000      # 3/110
device      bwtwo0 at obmem 7 csr 0xff000000 priority 4 # 3/60
device      bwtwo0 at obmem 8 csr 0x1000000      # 3/E

# 3/60 P4 color frame buffer overlay plane, or P4 monochrome frame buffer
device      bwtwo1 at obmem 7 csr 0xff300000 priority 4 # 3/60
# 3/60 plug-in color frame buffer overlay plane
device      bwtwo1 at obmem 7 csr 0xff400000      # 3/60

# 3/E plug-in color frame buffer overlay plane
device      bwtwo1 at obmem 8 csr 0xff400000      # 3/E

#
# Support for the GP/GP+/GP2 graphics processors.
# Requires cgtwo and/or cgnine.
#
device      gpone0 at vme24d16 ? csr 0x210000    # GP or GP+
device      gpone0 at vme24d32 ? csr 0x240000    # GP2

#
# Support for the Sun-2 color board, Sun-3 color board, or CG5 8-bit
# VME frame buffer.
#
device      cgtwo0 at vme24d16 ? csr 0x400000 priority 4
           vector cgtwointr 0xa8

#
# Support for the CG9 24-bit VME frame buffer.
#
device      cgnine0 at vme32d32 ? csr 0x08000000 priority 4
           vector cgnineintr 0xaa

#
# Support for the SunVideo board.
#
device      tvone0 at vme32d32 ? csr 0xa000000 priority 4
           vector tvoneintr 0xa9

#
# Support for the TAAC-1 Application Accelerator.
#
device      taac0 at vme32d32 ? csr 0x28000000

#
# Support for 2 Systech VPC-2200 line printer controllers.
#
device      vpc0 at vmel6d16 ? csr 0x480 priority 2 vector vpcintr 0x80
device      vpc1 at vmel6d16 ? csr 0x500 priority 2 vector vpcintr 0x81

#
# Support for the hardware Data Cipherring Processor (aka the DES chip).
# Suggested if you make heavy use of secure RPC or secure NFS.

```

```
#
device      des0 at obio ? csr 0x1c0000

#
# Support for the Floating Point Accelerator.
#
device      fpa0 at virtual ? csr 0xe0000000
```

9.4. How to Reconfigure the Kernel

This section explains how to reconfigure the kernel. Reconfiguring a kernel is easy to do, tailors the kernel to fit your environment, and significantly improves performance.

The SunOS 4.x GENERIC kernel supports many more devices than 3.x versions and is therefore much larger. To minimize the amount of memory occupied by the kernel, you must comment unnecessary kernel entries.

The main reasons to reconfigure the kernel are:

- To free memory that would otherwise be consumed by the unused kernel modules.
- To tell the kernel about hardware you added after the installation or software packages that require kernel modification and support.

Configuring a kernel is not a complex task. The GENERIC kernel configuration files contain instructions that help you determine which kernel entries are necessary for your system.

Kernel Reconfiguration Procedures

Note: In ALL cases, you must compile the kernel on a system of the same kernel architecture as the system on which it will be installed.

This section explains two ways to configure a custom kernel.

The kernel used by a diskless client can be built on its server *only* if the diskless client has the same kernel architecture as the server. In this case use Method 1.

If a diskless client has a *different* kernel architecture than the server, its kernel must be made on the client. In this case use Method 2.

Method 1:

You must have write privileges in the /usr file system of the system you are configuring to use this method to build a kernel for any of the following:

- A standalone system
- A server
- A diskless client (same kernel architecture as the server)
- A dataless client (same kernel architecture as the server)

If you do not have the necessary write privileges, use Method 2.

Use this method to build a kernel for any of the following:

- A diskless client (different kernel architecture than the server)
- A dataless client (different kernel architecture than the server)
- Diskless or dataless client without superuser privileges on the server

Background Information

You must be superuser or logged in as `root` to reconfigure a kernel.

`SYS_NAME`, as shown in the following example screens, represents the name given to the new kernel configuration file. You typically use the hostname of the system that will run the kernel. The name is automatically compiled into the kernel and is displayed when the kernel is booted. A kernel made from a configuration file named `MADDOG`, for example, announces itself when booted as follows:

```
SunOS Release 4.1 (MADDOG) #1: Wed Sep 14 15:33:16 PDT 1989
```

A kernel used by several client workstations of a server might be called `MADDOG_CLIENT`.

`CONFIG_FILE`, as shown in the following example screens, represents the name of the configuration file that you edit to customize your kernel. While `GENERIC` is the default configuration file (it contains all possible hardware and software supported by the operating system), it is easier to start by using a file similar to your hardware configuration needs. For example, use `DL` for diskless clients. Refer to *Parts of a Kernel Configuration File* earlier in this chapter.

The `ARCH` variable in the example screens show where to enter the implementation architecture of the workstation for which you are building a kernel.

Method 1

The following procedure shows you how to build a custom kernel for a local disk.

Step 1.

Create a kernel configuration file as `root` and execute the following:

```
# cd /usr/kvm/sys/k-arch/conf
# cp CONFIG_FILE SYS_NAME
# chmod +w SYS_NAME
# vi SYS_NAME

(Comment out unneeded kernel modules)

# config SYS_NAME
# cd ../SYS_NAME
# make
```

When the `make` is complete, install the kernel using the next step.

Step 2.

Install the configured kernel depending on which configuration of workstations the kernel was built on. The following three methods are presented to install a configured kernel.

- Standalone or server workstation
- Diskless client of the same kernel architecture as the server
- Dataless client of the same kernel architecture as the server

Standalone or Server

To install the kernel on a standalone system or a server, use the following method.

```
# mv /vmunix /vmunix.orig
# cp vmunix /vmunix
```

Diskless Client

To install the kernel on a diskless client of the same kernel architecture as the server, use the following method.

```
# mv /export/root/SYS_NAME/vmunix
/export/root/SYS_NAME/vmunix.orig
# cp vmunix /export/root/SYS_NAME/vmunix
```

Dataless Client

To install the kernel on a dataless client of the same architecture as the server, log onto the dataless workstation as `root` and use the following method.

```
# mv /vmunix /vmunix.orig
# cp /usr/kvm/sys/k-arch/SYS_NAME/vmunix /vmunix
```

Step 3.

Halt and reboot the system after installing the kernel as follows:

```
# /etc/halt
> b
```

Method 2

The following procedure shows you how to build a kernel for a diskless or dataless client on a server of a different architecture.

The following example makes the kernel in the `/sys` directory of the client. The client's `/sys` file is a symbolic link to `/usr/kvm/sys` and is located in the `/export/root` file system of the server.

You must have at least two megabytes of available disk space in order to make a kernel. If the space is not available in the `/export/root` file system, the procedure can be done in any other file system which has sufficient space *and* is writable by the client.

Step 1.

Create the configuration file by logging in a `root` and executing the

following:

```
# mkdir /home/hostname
# cd /home/hostname
# ln -s /usr/kvm/sys/* .
# rm k-arch
# mkdir k-arch
# cd k-arch
# ln -s /usr/kvm/sys/k-arch/* .
# rm conf
# mkdir conf
# cd conf
# ln -s /usr/kvm/sys/k-arch/conf/* .
```

Step 2.

Configure and make the kernel as follows:

```
# cp CONFIG_FILE SYS_NAME
# chmod +w SYS_NAME
# vi SYS_NAME

(Comment unneeded kernel modules)

# config SYS_NAME
# cd ../SYS_NAME
# make
```

Step 3.

Install the kernel after the make successfully completes as follows:

```
# mv /vmunix /vmunix.orig
# cp vmunix /vmunix
```

Step 4.

Halt and reboot the system after installing the kernel as follows:

```
# /etc/halt
>b
```

Step 5.

Clean up the file system after the client has successfully booted by saving a copy of your customized configuration file and removing `/home/hostname/k-arch` as follows:

```
# rm -rf /hostname/k-arch
```

9.5. Changing Swap Space

When you run SunInstall, by default it sets up swap space for an NFS server or standalone system on Partition B. In addition, it sets up the swap space for clients in `/export/swap/client_name` to enable the client machine to swap over NFS. At this time, you can specify the size of the swap partition and client swap files, or have SunInstall use the default.

After a system is in use, you may find the originally specified swap space for a machine is insufficient. Therefore, you will want to change swap size, either by increasing swap space on the disk or assigning an available partition on a first or second disk to include more swap space. To do this on a server, you have to back up all the server's file systems, then rerun SunInstall.

Procedures for Increasing Swap Space

To increase swap space when you need it for a diskless client, you can use a program called `mkfile`. The syntax is

```
# mkfile [-nv] size[k/b/m] filename ...
```

The option `n` tells `mkfile` to create an empty file. The option `v` selects verbose mode causing the system to display messages about what it is doing. The `size` argument specifies the size you want the file to be. The letters `k`, `b`, and `m` represent Kilobytes, Bytes, and Megabytes respectively. (Recommended swap size for a client machine is 16 Mbytes.) The `filename` argument, when configuring a client's swap space, should be the full pathname of the client machine's swap directory.

Here are steps you would take to increase swap space for diskless client `raks`.

Step 1.

Log in as superuser on your server machine.

Step 2.

You may want to check at this time to see how much swap space your client actually has. You can enter

```
# ls -l /export/swap client_name
-rw-----T 1 root      14680064 Jan 23 18:12 raks
```

and the number 14680064 shows that the client has about 14 Mbytes of swap.

Step 3.

Now, to increase the swap space, halt the system by entering

```
# /etc/halt
```

Step 4

Unexport the client's swap space as follows:

```
# exportfs -u /export/swap/raks
```

Step 5.

Then remove the client's swap space:

```
# rm /export/swap/raks
```

Step 6.

Now type the following:

```
# mkfile -n 16m /export/swap/raks
```

This creates an empty file of 16 Mbytes of size in /export/swap/raks.

Step 7.

You can again check the file to see how much swap space it now has:

```
# ls -l /export/swap/raks
-rw-----T 1 root      16777216 Jan 23 18:14 raks
```

Step 8.

Finally reboot client *raks*.

Here are the steps you would take to increase swap space on a server, standalone, or dataless client.

Step 1.

First determine if there is free disk space on your machine, if there is free space in /export/swap, do the following:

```
# mkfile -n 32m /export/swap/myswap
```

Step 2

Now add the following in your /etc/fstab file:

```
/export/swap/myswap swap swap ro 0 0
```

Step 3

Run swapon to specify additional device for swapping:

```
# swapon -a
```

Setting Up a Second Swap Partition

Another way you can increase swap size is to add or repartition an available partition on a first or second disk with another swap partition. Use the `format` program to repartition the disk. Then add the following to the server's /etc/fstab file:

```
/dev/disk_abbrevb /dev/disk_abbrev swap 0 0 0
```

where *disk_abbrev* represents the disk type and number of the second disk.

Then check the server's `/etc/rc` file for the line

```
swapon -a
```

Minimum Swap Space

If the amount of swap space your system has available is below the configurable minimum, you will receive a warning message. The minimum defaults to 512 blocks of 512 bytes each, or 256 Kbytes. If you wish to change this default, you can add the `SWAPWARN` option to your kernel configuration file. For example, to change the minimum swap space to 16 Mbytes, you would add the following options line to your configuration file:

```
options SWAPWARN=32768
```

After you reconfigure the new kernel, the system will give you a warning message if swap space falls below 16 Mbytes.

Maintaining Disks

`format` is a SunOS utility that enables you to format, label, repair, and analyze disks on your Sun system. Unlike other disk maintenance programs, `format` runs under the SunOS operating system. Because there are limitations to what can be done to the system disk while the operating system is running, `format` is also supported in a standalone UNIX environment. However, for most applications, running `format` under SunOS is sufficient and far more convenient.

`format` provides a friendly, menu based interface for disk maintenance. For most disk maintenance needs, there are instructions in this manual which can be followed to achieve the desired results. A detailed understanding of disk hardware is not necessary to perform basic maintenance functions.

10.1. An Overview of `format`

You use `format` to set up a disk for use by the operating system. Running `format` online provides many advantages, such as the ability to format many drives at once, and to save defect lists as normal ASCII files. This last feature is especially useful, as it allows you to recover when the defect list has become destroyed or corrupted. It is also possible (and occasionally necessary) to run `format` in a standalone fashion. For more details on this procedure, see *Installing SunOS 4.1*.

One factor remains constant between `format` and virtually all other disk programs - there is a *very real danger* destroying information inadvertently. When you format a drive, all the information on it is erased. For this reason, you should always back up your data before invoking `format`, especially when you first start using `format`.

You should familiarize yourself with a few key concepts used by the `format` utility before you actually use it.

Key concepts

The first concept is that of the *defect list*. This list is comprised of the known defective sectors on a disk, and is used during the formatting procedure to mark all defective sectors in the list as unusable. `format` requires a defect list before formatting a drive. Defect lists reside in a reserved area on a disk drive, so that they won't be destroyed during formatting. However, you should always save the information in your defect lists after doing a format operation as a precaution. You can do this by saving your defect lists as normal ASCII files with the `dump` command under `format`'s "Defect" menu. You should also photocopy and save the manufacturer's defect list that is usually attached to the drive when it is shipped. When it is necessary to read in a saved defect list, you use the `load`

command. Later sections in this chapter provide information and examples of what to do when there is no defect list or it has been corrupted.

`format` makes a necessary distinction between a *current* defect list and a *working* defect list. The *current* defect list is read in automatically when you select a current (default) disk, unless the disk is unformatted. When you need to make additions or deletions to the *current* defect list, `format` uses a temporary copy of this list called the *working defect list*. The working list is used as a temporary copy. When you are satisfied that your changes are valid, you update the current list with the `commit` command. Prior to invoking `commit`, you can undo all the changes you have made to the working list by using the `restore` command.

Another concept is `format`'s data file. By default, `format` looks in the file `/etc/format.dat` for the default disk types and partition tables. Having a data file external to the program makes it easy for you to add your own disk types or partition tables. If you wish to alter the default entries shipped by Sun, you should create alternate names of your own for them, to avoid any possible confusion.

Some of the `format` Pitfalls, and How to Avoid Them

Here is a list of actions you should or should not take when using `format`.

DO the following when using `format`:

- Back up all your files before doing anything else. Multiple copies are recommended.
- Save all your defect lists in files by using `format`'s `dump` command. The file name should include the drive type, model number, and serial number in the filename.
- Save the paper copies of the manufacturer's defect list that are shipped with your drive. These can be invaluable in case of disaster.
- Examine the command examples in the manual to help you through the procedures. Be especially careful when executing the `type`, `extract`, `commit`, and `format` operations.

DO NOT do the following when using `format`

- Run the `original` command under the `defect` menu when you have an unformatted SCSI disk. Due to a difference between SMD and SCSI drives, the `original` command may cause damage to a SCSI disk.
- Run the `extract` command under the `defect` menu when your disk is brand new or has a defect list, as you can destroy valid defect information.
- Change the Sun-supplied entries in the `format.dat` file. If you need slight variations on these, simply add new ones, being careful to give them unique names.
- Edit a defect list by hand. Doing so will invalidate the `checksum` and make the list useless.
- Do destructive surface analysis (that is, `write` or `compare` commands) on cylinder 0 or the last two cylinders. This would overwrite the label or defect

list, respectively.

- Do destructive surface analysis on a disk that has mounted partitions.

While this introduction has given you some of the basic concepts involved in the new `format` program, it is no substitute for reading the rest of this chapter. In most cases, you can use the examples in the text verbatim to achieve the desired results.

10.2. Interacting with `format`

This section describes the user interface of `format`. The types of input you need to specify are described, as well as some of the features that help you to interact with `format`.

Types of Input

Described below are the several types of input that `format` can require you to select.

Numbers

There are several places in `format` that require an integer as input. You must either specify the data or select one from a list of choices. In either case, the `help` facility causes `format` to print the upper and lower limits of the integer expected. Simply enter the number desired. The number is assumed to be in decimal unless a base is explicitly specified as part of the number (ie - `0x` for hexadecimal).

The following are examples of integer input.

```
Enter number of passes [2]: 34
Enter number of passes [34] 0xf
```

Block Numbers

Whenever you are required to specify a disk block number, there are a couple of ways to input the information.

You can specify the information as an integer representing the logical block number. As described above, you can specify the integer in any base, but the default is decimal. The maximum operator (a dollar sign, “\$”) can also be used here to let `format` select the appropriate value. Logical block format is used by the SunOS disk drivers in error messages. Some examples of this type of input are described later in this section.

The other way to specify a block number is the cylinder/head/sector format. In this format, you must specify explicitly the three logical components of the block number, the cylinder, head, and sector values. These values are still logical, but they allow you to define regions of the disk related to the layout of the media.

The specifics of the cylinder/head/sector format are as follows.

- Any number preceding the first slash is considered to be the cylinder number.
- Any number after the first slash but before the second slash is considered to be the head number.

- Any number following the second slash is considered to be the sector number.

If any of the numbers are not specified, the appropriate value is assumed to be zero. You can also use the maximum operator in place of any of the numbers and let `format` select the appropriate value. Below are some examples of this type of input.

```
Enter defective block number: 34/2/3
Enter defective block number: 23/1/
Enter defective block number: 457//
Enter defective block number: 12345
Enter defective block number: Oxabcd
Enter defective block number: 334/$/2
Enter defective block number: 892//§
```

Any time `format` prints a block number, it is printed in both of the above formats. Also, the `help` facility shows you the upper and lower bounds of the block number expected, in both formats.

Command Names

Command names are needed as input whenever `format` is sitting at a menu prompt. You can abbreviate the command names, as long as what is entered is sufficient to uniquely identify the command desired. You can also use the `help` facility to get a list of the commands available at any time.

Other Names

There are certain times in `format` when you must name something. In these cases, you are free to specify any string desired for the name. If the name has white space in it, the entire name must be enclosed in double quotes (""). Otherwise, only the first word of the name is used.

Supporting Features in `format`

Help Facility

`format` provides a help facility you can use any time `format` is expecting input. By simply entering a question mark `?`, you can request information about what is expected. `format` then prints a brief description of what type of input is needed. If appropriate, the legal boundaries of the input are also given. If you enter a question mark at a menu prompt, `format` reprints the list of commands that are available.

Maximum Operator

The maximum operator allows you to easily specify the largest legal number to `format`. Whenever you enter a block number, you can replace any of the components with a dollar sign `$`. This causes `format` to assign that component the largest value possible. This is especially useful when you specify a range of block numbers. For instance, if you want to specify all of cylinder 10 for analysis, you could enter the following:

```
Enter starting block number [6700, 10/0/0]: 10//
Enter ending block number [7369, 10/9/66]: 10/$/$
```

Defaults

`format` supports default values for input whenever possible. If `format` asks you to specify an object with a current value, the current value is always the default for the input. The default value is always displayed in square brackets ('[]') at the end of the input prompt. If you enter just a carriage return, and there is a default value, it is used by `format`.

10.3. Running `format`

The `format` utility runs under the SunOS operating system like any other utility. Whether running directly (after loading from tape,) or SunOS operating system, the command line used to run `format` is the same. The command line consists of the following parts:

```
format [options] [disk list]
```

The options all begin with a minus (-). Some of the options must be followed by a value. Note that several options can be grouped after a single minus, followed by the list of values for the options specified. As soon as `format` encounters something that is not an option or an option value, it assumes the disk list has begun. The rest of the command line is then assumed to be the disk list.

The following options are recognized by `format`:

- `-f command_file` Use the specified file for input instead of standard input. The file must contain commands in the same format that they would be entered through standard input, with the exception that no `continue?` messages are printed when running from a file.
- `-l log_file` Use the specified file to log the entire `format` session. All I/O to standard input, standard output, and error output are placed in the log file.
- `-x data_file` Use the specified file as the data file, instead of `format.dat`.
- `-d disk_name` Specifies which disk should be made current upon entry to the program. The disk is specified by its logical name (ie `-xy0`). This can also be accomplished by specifying a single disk in the disk list.
- `-t disk_type` Specifies the type of the disk made current upon entry to the program. The type is specified by its name as defined in the data file. This option can only be used if a disk is also being specified as described above.

`-p partition_name`

Specifies the partition table for the disk made current upon entry to the program. The table is specified by its name as defined in the data file. This option can only be used if a disk is being specified, and its type is either specified or available from the disk label.

`-s`

Run in silent mode. All prints to standard output are suppressed. Error messages are still printed. This is generally used in conjunction with the `-f` option when the commands have already been determined.

The disk list is used to tell `format` which disks should be searched for. If no disk list is specified, the list in the data file is used. Entries in the disk list are specified in logical format (ie - `xy0`). If a single entry is specified, that disk will automatically be made the current disk when `format` is entered.

As an example, to use `format` on a machine with two disks you can start `format` with the options:

```
# format -flxd /etc/myformat.cmds /tmp/format.mesgs /etc/myformat.dat xy1
```

This begins a non-interactive `format` session on the disk `xy1`. The commands to perform on that disk are taken from the file `/etc/myformat.cmds`, and the output is sent to `/tmp/format.mesgs`, rather than the console. Also, the data file `/etc/myformat.dat` is used instead of `/etc/format.dat`.

You should be aware that `format` requires that the raw devices for the disk to be present. If the `/dev/r???` device files for the disk you are operating on are not present, you must create them using `MAKEDEV`. See the chapter *Adding Hardware to Your System* and the `MAKEDEV(8)` man page for more information on using `MAKEDEV`.

10.4. Using `format` for Basic Maintenance

This section provides step-by-step instructions for using `format`. The most common uses for `format` are described in detail.

Often the first thing that you do with `format` is formatting the disk that is to be your system disk. Furthermore, every time you add a new disk to your system, you may need to run `format` in order to format the new disk. The procedures that you follow to actually format a disk differ depending on where the new disk came from.

Formatting a New Disk from Sun

Whenever you are adding a disk to your system that you purchased from Sun (including any disks that came with the system), you can follow these easy steps to ready the disk for use by the SunOS operating system. All disks shipped by Sun are formatted and labelled at the factory. This makes installation very simple. When you first attach a new disk to your system, Sun recommends that you reformat the entire drive. This insures that any head movement that occurred during shipment will not affect the performance of your new disk. Also, if you wish to partition your new disk differently than the Sun default partitions, you will have to create a partition table and relabel the disk. You can create the partition table from within `format`, or you can add it to the `format.dat` file. If you are running `format` directly from the release media, you must create the

partition from within `format`, or from within `SunInstall`. The following is an example of how you would install a new disk. In this example, you are adding `xy1` to the system. This example is running `format` under SunOS Release 4.1, not under `MUNIX`, so `xy1` can not be your system disk.

First, enter `format` and select the disk you wish to work on.

```

csh# format
Searching for disks...done

AVAILABLE DISK SELECTIONS:
  0. xd0 at xdc0 slave 0
      xd0: <CDC EMD 9720 cyl 1147 alt 2 hd 10 sec 48>
  1. xy1 at si0 slave 0
      0: <Fujitsu-M2312K cyl 587 alt 2 hd 7 sec 32>
Specify disk (enter its number): 1
selecting sd0: <Fujitsu-M2312K>
[disk formatted, defect list found]

FORMAT MENU:
  disk      - select a disk
  type      - select (define) a disk type
  partition - select (define) a partition table
  current   - describe the current disk
  format    - format and analyze the disk
  repair    - repair a defective sector
  show      - translate a disk address
  label     - write label to the disk
  analyze   - surface analysis
  defect    - defect list management
  backup    - search for backup labels
  quit

format>

```

At this point, note that the disk is already formatted and a defect list is present on it. This should be the case with disks shipped from Sun, except for the embedded SCSI disks of the newer Sun products. These embedded disks do not necessarily have a defect list when shipped. If the disk is not formatted, something is wrong and you should immediately exit `format` and use diagnostics to locate the problem. (How to do this will be explained later on in this chapter.) Now that the *current disk* is selected, you need to save a copy of the disk's defect list. If you are running directly from the installation media, see *Installing SunOS 4.1* for instructions for doing this. If you are bringing up the system disk, and working from within `SunInstall`, you should skip this part for now. Once you have Release 4.1 installed on the disk and running, you should then rerun `format` and execute just this part of the installation. This way you insure that you will always have a copy of the defect list handy. Note that before saving the defect list, you should compare it to the hard copy of the manufacturer's defect list. It should contain all the defects on the hard copy list. It may also contain some defects that were found at the Sun factory. If any of the defects are missing, you should add them to the defect list before proceeding.

```

format> defect
  restore - set working list = current list
  original - extract manufacturer's list from disk
  extract - extract working list from disk
  add - add defects to working list
  delete - delete a defect from working list
  print - display working list
  dump - dump working list to file
  load - load working list from file
  commit - set current list = working list
  quit

defect> print
  num      cyl      hd      bfi      len      sec
  1         11       6     31858      4
  2         21       4     14820      4
  3        106       7     27403      4
  .
  .
  .
  59        800       0      6156      4
  60        811       0     27738      6
  61        820       4      4502      3

total of 61 defects.
defect> dump
Enter name of defect file: 2333_defs.033537
defect file updated, total of 61 defects.
defect> q
format>

```

It is always a good idea to put the serial number and model of the disk in the name of the defect file. This lets you match up the defect file with the correct disk easily. If the serial number of the disk is not readily available, use some other unique identifier.

The next step is to reformat the *current disk*. The default values for the bounds of the `format` command will cause the entire drive to be reformatted. Also, by leaving the surface analysis parameters in their default state, 2 passes of analysis will be run over the disk when `format` completes. It is recommended that you do this analysis to verify the integrity of the media.


```

format> format
Ready to format. Formatting cannot be interrupted
and takes 11 minutes (estimated). Continue? y
Beginning format. The current time is Wed Feb 10 18:31:57 1988

Verifying media...
    pass 0 - pattern = 0xc6dec6de
    1217/11/14

    pass 1 - pattern = 0x6db6db6d
    1217/11/14
Total of 0 defective blocks repaired.

```

If any defects are found during surface analysis, they will be automatically repaired if possible. If the automatic repair did not succeed, you need to repair them manually. See the section on *@TitleOf(repair.sector)* for step-by-step instructions.

If you are happy with the default partitioning of the disk, you must now label the disk with the `label` command. The you are done with `format`. Your new disk is fully functional, and ready to be used by the system. If you wish to change the default partitioning of the disk, use the commands in the *partition menu* to create a table, and the `label` command to label the *current disk*.

Formatting a New Disk from Another Source

When you are adding a disk to your system that has never been formatted before, follow these easy steps to ready the disk for use by the operating system. All disks shipped by Sun are formatted at the factory, so this section applies only to disks that were purchased elsewhere. The following is an example of how you would install a new disk. In this example, you are adding `xd1` (a Fujitsu-M2333) to the system.

First, enter `format` and select the disk you wish to work on:

```
csht# format
Searching for disks...done

AVAILABLE DISK SELECTIONS:
  0. xd0 at xdc0 slave 0
     xd0: <CDC EMD 9720 cyl 1147 alt 2 hd 10 sec 48>
  1. xd1 at xdc0 slave 4
     xd1: <type unknown>

Specify disk (enter its number): 1

AVAILABLE DRIVE TYPES:
  0. Fujitsu-M2351 Eagle
  1. Fujitsu-M2333
  2. Fujitsu-M2361 Eagle
  3. CDC EMD 9720
  4. other

Specify disk type (enter its number): 1
selecting xd1: <Fujitsu-M2333>
[disk unformatted]
```

If the disk type is not one known to `format`, you can add support for the type in two ways. The easiest way is to add a definition to the `format.dat` file. This way the disk type will be known for all subsequent executions of `format`. Unfortunately, you can't do this if you are not running under SunOS. Thus, the second approach is designed to allow you to install your disk. By selecting `other` in the above `type` command, you can enter all the geometry information for the disk type by hand.

AVAILABLE DRIVE TYPES:

0. Micropolis 1355
1. Toshiba MK 156F
2. Micropolis 1558
3. Quantum ProDrive 80S
4. Quantum ProDrive 105S
5. CDC Wren IV 94171-344
6. SUN0104
7. SUN0121
8. SUN0207
9. SUN0327
10. SUN0669
11. Sun1.0G
12. other

```
Specify disk type (enter its number) [0]: 12
Enter number of data cylinders: 1018    Note: This info is for example
Enter number of alternate cylinders [2]: <CR>  only-- use the info supplied
Enter number of physical cylinders [1020]: <CR>  with your disk
Enter number of heads: 8
Enter number of data sectors/track: 34
Enter rpm of drive [3600]: <CR>
Enter disk type name (remember quotes): "Hyper THX-1128"

format>
```

Now the disk type will then be known for this instance of `format`. If you have to run `format` again on this disk however, you will have to enter the information again. Thus, it is always best to add the disk types to the `format.dat` file whenever possible. To add an entry to the `format.dat` file, refer to the later section called *The format.dat File*.

Now that the *current disk* and *current disk type* are set, the “defect list” can be created. A defect list is necessary so that a disk can avoid using blocks which are defective. A “defect list,” can be made in one of several ways. If you are using a disk controller like the Xylogics 7053 which supports the `original` command, then `original` should be used to create the list. If you are using a MD21 controller (which doesn’t support the `original` command,) then the `add` command must be used to enter the defect list by hand.

Note: A SCSI disk uses a different type of block addressing than other disks. Whenever you are working with block addresses for a SCSI disk, always use *bytes-from-index* (bfi) addressing.

A hard copy of the defect list should be attached to the disk. This can be used to enter the list by hand, in the event that all else fails. There is one other exception to the above rules. The Emulex MD21 controller supports the `original` command, but only after the *current disk* has been formatted. Thus, if you are adding a disk to this controller, you should skip this step for now and come back to it after you have formatted the disk. In this example, you are using the Xylogics 7053 controller, which supports the `original` command for brand new disks:

```

format> defect
  restore - set working list = current list
  original - extract manufacturer's list from disk
  extract - extract working list from disk
  add      - add defects to working list
  delete  - delete a defect from working list
  print   - display working list
  dump    - dump working list to file
  load    - load working list from file
  commit  - set current list = working list
  quit

defect> original
Ready to update working list. This cannot be interrupted
and may take a long while. Continue? y
Extracting manufacturer's defect list.
Extraction complete.
Working list updated, total of 61 defects.
defect>

```

After initializing the *working defect list* with either `original` or `add`, you should compare the resulting list with the hard copy supplied with the disk. To do this you should use the `print` option from the `:defect` menu

```

defect> print
  num      cyl      hd      bfi      len      sec
  1         11       6      31858     4
  2         21       4      14820     4
  3         106      7      27403     4
  .
  .
  .
  59        800      0       6156     4
  60        811      0      27738     6
  61        820      4       4502     3
total of 61 defects.
defect> commit
ready to update Current Defect List, continue? y
Current Defect List updated, total of 61 defects.
Disk must be reformatted for changes to take effect.
defect>

```

If it doesn't match, you can use the `add` and `delete` commands to make them match. Once the *working defect list* is correct, you must run the `commit` command to tell `format` you have a complete defect list. The *working defect list* is then copied to the *current defect list*, which is used when the *current disk* is formatted.

Before exiting the `defect` menu, you should save the defect list onto disk or tape. If you are not running under SunOS see *Installing SunOS 4.1* for

instructions for doing this. When the system is installed and the operating system is running, rerun just this part of `format` to save the defect list. It is always a good idea to put the serial number and model of the disk in the name of the defect file. This lets you match up the defect file with the correct disk easily. If the serial number of the disk is not readily available, use some other unique identifier. Once you have saved the defect list, you can quit the *defect menu*.

```
defect> dump
Enter name of defect file: 2333_defs.033537
defect file updated, total of 61 defects.
defect> q
format>
```

The next step is to format the *current disk*. The default values for the bounds of the `format` command will cause the entire drive to be formatted. Also, by leaving the surface analysis parameters in their default state, two passes of analysis will be run over the disk when the format completes. It is recommended that you do this analysis to verify the integrity of the media.

```
format> format
Enter starting block number [0, 0/0/0]: <cr>
Enter ending block number [551409, 822/9/66]: <cr>

Ready to format. Formatting cannot be interrupted
and takes 11 minutes (estimated). Continue? y
Beginning format. The current time is Wed Feb 10 18:31:57 1988

Verifying media...
    pass 0 - pattern = 0xc6dec6de
    1217/11/14

    pass 1 - pattern = 0x6db6db6d
    1217/11/14
Total of 0 defective blocks repaired.
```

If any defects are found during the surface analysis, they will be automatically repaired if possible. If the automatic repair did not succeed, you need to repair them manually. See the section on *@TitleOf(repair.sector)* for step-by-step instructions.

If the disk has a partition table defined in the `format.dat` file, and you are happy with using this default table, you are ready to label the disk. Note that if there are multiple partition tables in the `format.dat` file for a given disk type, the first one in the file is always used as the default. If there are no predefined tables, or you want to alter the table for this specific disk, you need to use the *partition menu* commands to create your own partition table. Once you are ready to label the disk, you simply run the `label` command. In this example, you are using the default partition table, so you do not have to specify one.

```
format> label
Current Partition Table is not set, using default.
Ready to label disk, continue? y
```

Once the disk is labeled, you are done with `format`. Your new disk is fully formatted, and ready to have a file system put on it using `newfs`.

Setting up or Changing Disk Partitions

Whenever you set up a new disk, it might be necessary or desirable to partition the disk into separate regions, called *partitions*. This is also necessary in order to install some software packages. Partitioning a disk is done after `format`, and in order to repartition a disk the disk will need to be relabeled.

The process of partitioning a disk is very simple. First, back up all information on the disk. Repartitioning can cause some filesystem contents to be inaccessible. Then, unmount the disk if you are repartitioning a currently active disk. Now, enter `format` and select the disk.

Now select the partition command in order to repartition the disk.

```
format> partition

PARTITION MENU:
  a      - change 'a' partition
  b      - change 'b' partition
  c      - change 'c' partition
  d      - change 'd' partition
  e      - change 'e' partition
  f      - change 'f' partition
  g      - change 'g' partition
  h      - change 'h' partition
  select - select a predefined table
  name   - name the current table
  print  - display the current table
  label  - write partition map and label to the disk
  quit
partition>
```

If you want to use a predefined partition table, you can select that table using the `select` command. This is useful if you wish to set an old disk back to the original partition sizing, or if you have defined a partition table previously and you wish to set up according to that table. In this example you don't wish to do that, so you must create a new table. You will do this by modifying the current table to reflect the new sizes that you want. The manner in which this is done is you select the partition that you wish to define, and then you enter the cylinder in which that partition will start, followed by the size of the cylinder. In this example, you wish to add a new partition "f" to an existing disk that is not to be a system disk. Print the current table first to see what the current partition sizing is:

```

partition> print
Current partition table (original xyl):
  partition a - starting cyl      0, # blocks      0 (0/0/0)
  partition b - starting cyl      0, # blocks      0 (0/0/0)
  partition c - starting cyl      0, # blocks    550070 (821/0/0)
  partition d - starting cyl      0, # blocks      0 (0/0/0)
  partition e - starting cyl      0, # blocks      0 (0/0/0)
  partition f - starting cyl      0, # blocks      0 (0/0/0)
  partition g - starting cyl      0, # blocks    550070 (821/0/0)
  partition h - starting cyl      0, # blocks      0 (0/0/0)
partition>

```

The current sizing has the whole disk being defined as the “g” partition. To create a new partition, you have to decrease the size of the “g” partition while increasing the size of the “f” partition. First, decide how big you want to make the new partition. If the software that you are installing requires 4 Megabytes of disk space, then the partition will have to be large enough to hold that. Now you need to figure out how many disk blocks will be needed to give you that much space. Since SunOS disk blocks are about 512 bytes in size, you need 8192 blocks of space. The problem now comes about that partitions must always start at the beginning of a cylinder. So what is the minimum number of cylinders that will give you 8192 blocks? The exact number is dependent on the geometry of the disk. You can determine this number by dividing the number of blocks in the disk as shown for the “c” partition by the number of cylinders on the disk. In this example, the Fujitsu drive uses $(550070 / 821)$ or 670 blocks per cylinder, so the number of needed cylinders is $(\text{number of needed blocks} / \text{number of blocks per cylinder})$ or $(8192 / 670)$. As this number is slightly higher than 12, we must use 13 cylinders. Alternatively, if you are not using the MD21 controller, you can also determine this value using the `show` command from the `format` menu. Simply enter the number of disk blocks needed, and `format` will translate this into the necessary number of blocks. So for the above example:

```

partition> quit

FORMAT MENU:
  disk      - select a disk
  type      - select (define) a disk type
  partition - select (define) a partition table
  current   - describe the current disk
  format    - format and analyze the disk
  repair    - repair a defective sector
  show      - translate a disk address
  label     - write label to the disk
  analyze   - surface analysis
  defect    - defect list management
  backup    - search for backup labels
  quit

format> show
Enter a disk block: 8192
Disk block = 8192 = 0x2000 = (12/0/32)

```

Now, set the “f” partition to start at the beginning of the disk, end 31 cylinders into the disk, and have the “g” partition take of the rest of the space:

```

partition> f

partition f - starting cyl 0, # blocks 0 (0/0/0)

Enter new starting cyl [0]: 0
Enter new # blocks [0, 0/0/0]: 13/0/0
partition> g

partition g - starting cyl 0, # blocks 550070 (821/0/0)

Enter new starting cyl [0]: 13
Enter new # blocks [550070, 821/0/0]: 808/0/0
partition>

```

Now all that remains to be done is to label the disk with the new partition information using the “label” command from the `partition>` menu. Of course, to actually make use of the partition you just created, you will need to run `newfs` on it, which is covered in the *File Maintenance* chapter.

Repairing a Defective Sector

If a disk on your system has a defective sector, you can repair the sector using the step by step instructions given in this section. You may become aware of defective sectors through several different means. If you run surface analysis on a disk, and repairing is not supported by the controller or is disabled, any defects found will simply be reported to you. In these cases, the exact sector in error will be known, since surface analysis very carefully pinpoints the source of the error before notifying you. In the course of running the operating system, you may get a number of error messages from the disk driver concerning a particular portion

of the disk. If these errors seem media related, you may wish to look for a defective sector in that area. Note that you should not take the sector number reported by the operating system as gospel. Since the system does disk operations many sectors at a time, it is often hard to pinpoint exactly which sector caused a given error. You should always try to use the analysis tools provided in `format` to find the defective sector. For this example, assume that you got several ECC errors for sector number 12345 on `xd0`. First, unmount the affected partition using the `umount` command, if possible. Now, enter `format` and select the disk.

```

csh# format
Searching for disks...done

AVAILABLE DISK SELECTIONS:
  0. xd0 at xdc0 slave 0
      xd0: <CDC EMD 9720 cyl 1147 alt 2 hd 10 sec 48>
Specify disk (enter its number): 0
selecting xd0: <CDC EMD 9720 cyl 1147 alt 2 hd 10 sec 48>
[disk formatted, defect list found]
Warning: Current Disk has mounted partitions.

FORMAT MENU:
  disk      - select a disk
  type      - select (define) a disk type
  partition - select (define) a partition table
  current   - describe the current disk
  format    - format and analyze the disk
  repair    - repair a defective sector
  show      - translate a disk address
  label     - write label to the disk
  analyze   - surface analysis
  defect    - defect list management
  backup    - search for backup labels
  quit

format>

```

Next, you need to pinpoint exactly where the defective sector is. Because the `xd0` disk is being used by the operating system, run the `read` test. This will not interfere with the operation of the system. Run the `read` test over a few tracks worth of sectors, using the sector number given by the SunOS error message as the midpoint of your search. Also set up the analysis parameters to loop continuously, since it is important to find the problem sector. This will cause `format` to run analysis until you stop it with a **CTRL-C**. Also, run the `read` test with a transfer size of 1 sector, so it will be very clear which sector caused any errors encountered. If the analysis fails to show any defects, you may want to repair the sector given in the SunOS error message anyway, in hopes that it is the defective sector. Or, you may want to wait a while and see if the error was transient. However, for this example, assume that the error showed itself during analysis.

```

analyze> setup
Analyze entire disk [no]? n
Enter starting block number [0, 0/0/0]: 12330
Enter ending block number [584159, 1216/9/47]: 12360
Loop continuously [no]? y
Repair defective blocks [yes]? n
Stop after first error [no]? <cr>
Use random bit patterns [no]? <cr>
Enter number of blocks per transfer [31, 0/0/31]: 1
Verify media after formatting [yes]? <cr>
Enable extended messages [no]? <cr>
Restore defect list [yes]? <cr>
Create defect label [yes]? <cr>

analyze> read
Ready to analyze (won't harm SunOS). This takes a long time,
but is interruptible with CTRL-C. Continue? y

    pass 0
    25/7/24

    pass 1
Block 12354 (18/4/18), Corrected media error (hard data ecc)
    25/7/24

    pass 3
Block 12354 (18/4/18), Corrected media error (hard data ecc)
^C 25/7/24

Total of 0 defective blocks repaired.
analyze> quit

```

Because some controllers do not support repairing, the read test above was run with automatic repair turned off. If we wanted to, we could have had `format` repair the defective sector as soon as it was found. However, we will instead show the two methods for repairing a defective sector manually. If the controller does support repairing, it is a simple case of running the `repair` command on the sector that surface analysis showed to be bad. This will preserve the data in the bad sector if possible, so there is no need to back up your disk.

```

format> repair
Enter defective block number: 18/4/18
Ready to repair defect, continue? y
Repairing 18/4/18.
Repair succeeded.
format> quit
csh#

```

However, if repairing is not supported, it is a little more involved. First, the defective sector must be added to the *working defect list*. Also, the *working defect list* must then be committed to the *current defect list* so the new defect will show up on the disk's copy of the defect list.

Note: Always use bfi addressing when working with SCSI disks.

```
format> defect
  restore - set working list = current list
  original - extract manufacturer's list from disk
  extract - extract working list from disk
  add      - add defects to working list
  delete  - delete a defect from working list
  print   - display working list
  dump    - dump working list to file
  load    - load working list from file
  commit  - set current list = working list
  quit

defect> add
  0. bytes-from-index
  1. logical block
Select input format (enter its number) [0]: 1
Enter defective block number: 18/4/18
  num      cyl      hd      bfi      len      sec
   62      18      4       4       18      18
ready to add defect, continue? y
defect number 62 added.
Enter defective block number: ^C
defect> commit
ready to update Current Defect List, continue? y
Current Defect List updated, total of 62 defects.
Disk must be reformatted for changes to take effect.
defect> quit
format>
```

Finally, the *current disk* must be reformatted, so the media itself reflects the new defect. If possible, you can reformat only the portion of the *current disk* that the defect lies in. However, most controllers that do not support repairing also require that you format the entire disk at once, so you will probably need to format the whole disk. Remember to back up your disk before doing any format operation. In this example, only the portion of the *current disk* necessary is reformatted.

```
format> format
Enter starting block number [0, 0/0/0]: 18/4/0
Enter ending block number [551409, 822/9/66]: 18/4/$

Ready to format. Formatting cannot be interrupted
and takes 1 minutes (estimated). Continue? y
Beginning format. The current time is Wed Feb 10 18:31:57 1988

Verifying media...
    pass 0 - pattern = 0xc6dec6de
18/4/47

    pass 1 - pattern = 0x6db5db6d
18/4/47

.
.
.

^C

Total of 0 defective blocks repaired.
format> quit
```

***NOTE** Because you previously set the surface analysis parameters to loop continuously, you had to use **CTRL-C** to get out of the analysis that runs after you format. The sector is now repaired, and the disk is fully functional once again.*

Relabeling a Corrupted Disk

If a disk on your system is corrupted, and the primary label is destroyed, you may be able to restore the label from backup copies that are stored on the disk. If this doesn't work, you will have to recreate the partition table yourself and relabel the disk. If the disk used a predefined partition table, this is not difficult. However, if the partition table was not defined in `format.dat`, you will have to remember the partition information that was on the disk. This is why you should always add the partition tables for all your disks to the `format.dat` file. The following is an example of how you would attempt to restore the label. These commands can all be done while running `format` under `MUNIX`, so this procedure will even work if your system disk has been corrupted. In this example, you are trying to relabel `xd0`.

First, enter `format` and select the disk you need to work on:

```

csh# format
Searching for disks...done

AVAILABLE DISK SELECTIONS:
    0. xd0 at xdc0 slave 0
       xd0: <drive type unknown>
Specify disk (enter its number): 0

AVAILABLE DRIVE TYPES:
    0. Fujitsu-M2351 Eagle
    1. Fujitsu-M2333
    2. Fujitsu-M2361 Eagle
    3. CDC EMD 9720
    4. other
Specify disk type (enter its number): 1
selecting xd0: <Fujitsu-M2333>
[disk formatted, defect list found]

FORMAT MENU:
disk          - select a disk
type          - select (define) a disk type
partition    - select (define) a partition table
current      - describe the current disk
format       - format and analyze the disk
repair       - repair a defective sector
show         - translate a disk address
label        - write label to the disk
analyze      - surface analysis
defect       - defect list management
backup       - search for backup labels
quit

format>

```

Once the disk type is specified, format knows the geometry and is now able to locate the defect list. If the defect list is not found at this point (and you know the disk had a defect list), it was probably also corrupted. You can recreate the defect list after you are done fixing the label. See the section on *@TitleOf(defect.list)*. At this point, you can search for the backup labels. Simply execute the backup command. If a backup label is found, format will use it to rewrite the primary label. The disk is then functional again.

```

format> backup
Searching for backup labels...found.
Restoring primary label and defect list.
format>

```

If no backup labels were found, you will have to reconstruct the label by hand. If the partition table that was on the disk is in the `format.dat` file, this is fairly easy. If you did not use a predefined partition table, it is more complicated. This is why you should add the partition tables for all the disks on your system to the `format.dat` file.

For disks that use a predefined partition table, creating the label by hand is simple. The geometry information is already known, since the `type` command was run, so it is only the partition information that must still be entered. This is done via the `select` command. First, the *partition menu* must be entered. Next, the `select` command is run. The desired partition table can then be chosen from the list of possibilities.

```
format> partition

PARTITION MENU:
a      - change 'a' partition
b      - change 'b' partition
c      - change 'c' partition
d      - change 'd' partition
e      - change 'e' partition
f      - change 'f' partition
g      - change 'g' partition
h      - change 'h' partition
select - select a predefined table
name   - name the current table
print  - display the current table
label  - write partition map and label to the disk
quit

partition> select
0. Fujitsu-M2333
1. my table
Specify table (enter its number) [0]: 1

partition> label
Ready to label disk, continue? y
```

Once you have selected the correct partition table, all that is left is to label the current disk. You can use the `label` command to label the disk.

Creating a Defect List

There are certain situations under which you will need create a defect list for a disk on your system. All Sun disks shipped before SunOS 3.2 did not have defect lists on them. If you wish to operate on a disk of this type, you will need to create a defect list first. Also, the defect list on a disk may be corrupted by a media problem, or some other catastrophe. The instructions in this section explain how to recreate the defect list for a disk on your system.

No matter what the reason for needing the defect list, there are only a couple of approaches to solving the problem. First, you need to enter `format` and select the disk that needs work.

```

csh# format
Searching for disks...done

AVAILABLE DISK SELECTIONS:
    0. xd0 at xdc0 slave 0
        xd0: <Fujitsu-M2333 cyl 821 alt 2 hd 10 sec 67>
Specify disk (enter its number): 0
selecting xd0: <Fujitsu-M2333>
[disk formatted, no defect list found]

FORMAT MENU:
    disk          - select a disk
    type          - select (define) a disk type
    partition     - select (define) a partition table
    current       - describe the current disk
    format        - format and analyze the disk
    repair        - repair a defective sector
    show          - translate a disk address
    label         - write label to the disk
    analyze       - surface analysis
    defect        - defect list management
    backup        - search for backup labels
    quit

format>

```

Note that you can tell the defect list was not present on the disk by the message printed in the square brackets. Here is where the two approaches differ. If your disk once had a defect list on it, and you had saved a copy of that list in a SunOS file with the `dump` command, recreating the defect list is very easy. You simply enter the *defect menu*, and use the `load` command to initialize the *working defect list*. If you can initialize the *working defect list* this way, you can skip the next section and go on to where you commit the defect list.

```

format> defect
    restore      - set working list = current list
    original-    - extract manufacturer's list from disk
    extract      - extract working list from disk
    add          - add defects to working list
    delete       - delete a defect from working list
    print        - display working list
    dump         - dump working list to file
    load         - load working list from file
    commit       - set current list = working list
    quit

defect> load
Enter name of defect file: 2333_defs.033537
ready to update working list, continue? y
working list updated, total of 60 defects.
defect>

```

If you do not have a copy of the defect list in a SunOS file, you can try to extract the defect list from the disk media. The `extract` command actually looks at the data on the *current disk* and calculates which sectors have been repaired. This lets it create a defect list that represents the defects on the disk. Simply enter the *defect menu* and run the `extract` command. This command can take quite a while, so be prepared to wait. If you initialize the *working defect list* this way, you should always immediately run the `dump` command to save a copy of the defect list in a SunOS file. This will prevent you from ever having to extract again later.

```
format> defect
  restore - set working list = current list
  original - extract manufacturer's list from disk
  extract - extract working list from disk
  add      - add defects to working list
  delete   - delete a defect from working list
  print    - display working list
  dump     - dump working list to file
  load     - load working list from file
  commit   - set current list = working list
  quit

defect> extract
Ready to update working list. This cannot be interrupted
and may take a long while. Continue? y
Extracting current defect list.
Extraction complete.
Working list updated, total of 60 defects.
defect> dump
Enter name of defect file: 2333_defs.033537
defect file updated, total of 60 defects.
defect>
```

Some controllers (such as the MD21) do not support extracting the defect list this way. If this is the case, the only way to create a defect list for the disk is to enter the defects manually with the `add` command.

Now, assume that the *working defect list* has been initialized using one of the above methods. Next, you must commit the *working defect list* to the *current defect list*. This causes it to be used when the disk is operated on.

```
defect> commit
ready to update Current Defect List, continue? y
Current Defect List updated, total of 60 defects.
Disk must be reformatted for changes to take effect.
defect> quit
format>
```

After committing the defect list, all that is necessary is to get a copy of the defect list onto the *current disk*. If you were creating the defect list because you intend to repair or reformat the *current disk*, then those commands will write the defect list onto the *current disk*. However, if you were creating the defect list just to

have one (which is a good idea), you must manually write it to the *current disk*. The easiest way to do this is with the `backup` command. By running the `backup` command and forcing it to relabel the disk, you also cause it to put the defect list back on the disk. This way, the *current disk* will have a defect list that reflects the state of the disk media. You should be careful when you run the `backup` command, since `format` cannot enforce that the defect list you are writing to the disk really reflects the state of the disk media. Be sure you initialized the *current defect list* correctly before proceeding. When you run the `backup` command, it will notify you that the *current disk* was already labeled. That's ok, force it to relabel the disk anyway. This is the only way to cause the defect list to be written out.

```
format> backup
Disk had a primary label, still continue? y
Searching for backup labels...found.
Restoring primary label and defect list.
format> quit
csh%
```

The *current disk* now has the correct defect list on it, and can be used by `format` without incident.

10.5. The `format.dat` File

The `format` data file allows you to configure `format` to support your specific system. There are three things that can be defined in the data file -- search paths, disk types, and partition tables. Usually, the data file shipped with Sun systems is sufficient. It contains all the information necessary to support standard Sun disks. You will want to modify the data file for your system if you have one of the following:

- A disk that has a unit number or controller number that is not found in the `GENERIC` configuration file.
- A disk that is a different model than those supported by Sun.
- A disk with a partition table that is different from the table Sun shipped it with.

Even if one of the above criteria is met, it is possible to run `format` without modifying the data file. However, certain information would have to be entered while `format` is running. By modifying the data file to support your configuration, the information will automatically be known to `format`.

There are several ways to specify the location of your data file to `format`. If a pathname is given with the `-x` command line option, that file is always used as the data file. If the `-x` option is not specified, then `format` looks in the current directory for a file named `format.dat`. If the file exists, it is then used as the data file. If neither of these methods yields a data file, then `format` uses the file `/etc/format.dat` as the data file. This file is shipped with your Sun system, so it should always be present.

The data file contains definitions that are read in by `format` when it starts up. Each definition starts with a keyword, showing which of the three types of definitions it is. The three legal keywords in the data file are `search_path`, `disk_type`, and `partition`.

Use the `search_path` keyword to tell `format` which disks it should search for when it starts up. The list in the default data file contains all the disks in the `GENERIC` configuration file. If your system has disks that are not in the `GENERIC` configuration file, you should add them to the `search_path` definition in your data file. Unlike the other definitions, the data file can contain only one `search_path` definition. However, this single definition lets you specify all the disks you have in your system.

Use the `disk_type` keyword to define a specific controller and disk model so `format` can operate on such a disk. Each `disk_type` definition contains information concerning the physical geometry of the disk. The default data file contains definitions for the controllers and disks that Sun supports. You only need to add a new `disk_type` if you have a non-Sun disk. You can add as many `disk_type` definitions to the data file as you want.

Use the `partition` keyword to define a partition table for a specific disk type. It contains the partitioning information, plus a name that lets you refer to it in `format`.

The default data file contains partition definitions for all the partition tables that Sun ships disks with. You should add a partition definition if you repartitioned any of the disks on your system. You can add as many partition definitions to the data file as you require.

The syntax of the data file is an expanded version of the termcap syntax. The following rules apply to the data file:

- The hash sign ('#') is the comment character. Any text on a line after a hash sign is not interpreted by `format`.
- The data file consists of a series of definitions. Each definition appears on a single logical line. This means that if you break the definition across multiple lines in the data file, all but the last line of the definition must end with a backslash ('\'). A carriage return that is not preceded by a backslash indicates the end of a definition.
- A definition consists of a series of assignments. Each assignment has an identifier on the left side and a value(s) on the right side. The assignment operator is the equal sign ('='). The assignments within a definition must be separated by a colon (':').
- White space is ignored by `format`. You can use it to make the data file more readable, it will all be stripped off. If you want an assignment value to actually contain white space, you can enclose the entire value in double quotes (""). This will cause the white space within the quotes to be preserved as part of the assignment value. Because of this feature, it is a good idea to enclose all string values in double quotes, in case you want them to contain white space.

- Some assignments can have multiple values on the right hand side. When there are multiple values, they must be separated by a comma (',').

Search Path

The `search_path` definition consists of only one assignment. The keyword itself is assigned a list of disk names. The disk names are as they appear in the boot messages. Here is an example of the `search_path` definition:

```
#
# This is the search path for format. It contains all the disks that
# will be searched for if no disk list is given on the command line.
#
search_path = xy0, xy1, xy2, xy3, xd0, xd1, xd2, xd3, xd4, xd5, xd6, \
             xd7, xd8, xd9, xd10, xd11, xd12, xd13, xd14, xd15, sd0, sd1, sd2, sd3
```

Disk Type

A `disk_type` definition consists of several general assignments and some assignments that are controller specific. The keyword itself is assigned the name of the disk type. This name appears in the disk's label, and is used to identify the disk type whenever `format` is run. Remember that you should enclose the name in double quotes so any white space in the name is preserved. The following identifiers must also be assigned values in all `disk_type` definitions:

- `ctlr` - must be assigned a value that represents the controller type the disk type can be attached to. Currently, the supported values for this assignment are XY450 for Xylogics 450/451 controllers, XD7053 for Xylogics 7053 controllers, MD21 for Emulex MD21 controllers (and embedded SCSI disks), and ACB4000 for Adaptec ACB4000 controllers.
- `ncyl` - must be assigned the number of data cylinders in the disk type. This determines how many logical cylinders of the disk the system will be allowed to access.
- `acyl` - must be assigned the number of alternate cylinders in the disk type. These cylinders are used by `format` to store information such as the defect list for the drive. You should always leave at least two cylinders for alternates.
- `pcyl` - must be assigned the number of physical cylinders in the disk type. This number is used to calculate the boundaries of the disk media. This number is usually equal to `ncyl` plus `acyl`, but there are some circumstances under which it is not. For instance, the Emulex MD21 controller requires four cylinders for internal controller use, so they must be left off the other assignments. Also, to make disks field replaceable with second sources, some disks are artificially limited to be the same size as another type.
- `nhead` - must be assigned the number of heads in the disk type. This number is used to calculate the boundaries of the disk media.
- `nsect` - must be assigned the number of data sectors per track in the disk type. This number is used to calculate the boundaries of the disk media. Note

that this is only the data sectors, any spares are not reflected in the assignment.

- `rpm` - must be assigned the rotations per minute of the disk type. This information is put in the label and later used by the file system to calculate the optimal placement of file data.
- `bpt` - must be assigned the physical number of bytes per track for the disk type. This number is used to calculate the boundaries for defects that are in bytes from index format.

Other assignments may be necessary depending on which controller the disk type is attached to. For XY450/451 controllers, the following assignments are also required:

- `bps` - must be assigned the total number of bytes per sector, including the header and gaps, in the disk type. This number is necessary to locate defects within a track. See the disk manual for information on how to calculate this number.
- `drive_type` - must be assigned the drive type of the disk type. The drive type is a number between 0 and 3 that the 450/451 controller uses to identify the disk's geometry. See the controller manual for more information.

For XD7053 controllers, the following assignments are also required:

- `bps` - must be assigned the total number of bytes per sector, including the header and gaps, in the disk type. This number is necessary to locate defects within a track. See the disk manual for information on how to calculate this number.

For MD21 controllers, the following assignments are also required:

- `fmt_time` - must be assigned a number indicating how long it takes to format a given drive. See the controller manual for more information.
- `cache` - must be assigned a number that controls the operation of the onboard cache while format is operating. See the controller manual for more information.
- `trks_zone` - must be assigned a number that specified how many tracks you have per defect zone, to be used in alternate sector mapping. See the controller manual for more information.
- `asect` - the number assigned to this parameter specifies how many sectors are available for alternate mapping within a given defect zone. See the controller manual for more information.

For ACB4000 controllers, the following assignments are also required:

- `skew` - must be assigned the buffer skew for the disk type.
- `precomp` - cylinder at which to begin write precompensation.

Below are some examples of `disk_type` definitions.

```

disk_type = "Fujitsu-M2361 Eagle" \
: ctrlr = XY450 : fmt_time = 4 \
: ncy1 = 840 : acyl = 2 : pcy1 = 842 : nhead = 20 : nsect = 67 \
: rpm = 3600 : bpt = 40960 : bps = 600 : drive_type = 3

disk_type = "CDC EMD 9720" \
: ctrlr = XD7053 \
: ncy1 = 1147 : acyl = 2 : pcy1 = 1217 : nhead = 10 : nsect = 48 \
: rpm = 3600 : bpt = 30240 : bps = 613

disk_type = "Micropolis 1355" \
: ctrlr = MD21 \
: ncy1 = 1018 : acyl = 2 : pcy1 = 1024 : nhead = 8 : nsect = 34 \
: rpm = 3600 : bpt = 20832

disk_type = "Fujitsu M2243As" \
: ctrlr = ACB4000 \
: ncy1 = 752 : acyl = 2 : pcy1 = 754 : nhead = 11 : nsect = 17 \
: rpm = 3600 : bpt = 10416 : skew = 2 : precomp = 754

```

Partition Tables

A partition definition contains some general assignments, and some optional assignments. The keyword itself is assigned the name of the partition table. This name is used inside `format` to identify the table. Remember to enclose the name in double quotes so any white space in the name is preserved. The following identifiers must also be assigned values in all partition definitions:

- `disk` - must be assigned the name of the `disk_type` that this partition table is defined for. This name must appear exactly as it does in the `disk_type` definition.
- `ctrlr` - must be assigned a value that represents the controller type disks with this partition table can be attached to. Currently, the supported values for this assignment are XY450 for Xylogics 450/451 controllers, XD7053 for Xylogics 7053 controllers, MD21 for Emulex MD21 controllers, and ACB4000 for Adaptec ACB4000 controllers. The controller type specified here must also be defined for the `disk_type` chosen above.

The other assignments in a partition definition describe the actual partition information. The identifiers are the letters `a` through `h`, with each letter representing that partition. These assignments are optional; any partition not explicitly assigned is set to 0 length. The value of each of these assignments is a pair of numbers separated by a comma. The first number is the starting cylinder for the partition, and the second is the number of sectors in the partition. Below are some examples of partition definitions.

```

partition = "Fujitsu-M2351 Eagle" \
: disk = "Fujitsu-M2351 Eagle" : ctrlr = XY450 \
: a = 0, 16560 : b = 18, 34040 : c = 0, 772800 : g = 55, 722200

partition = "Fujitsu-M2351 Eagle Old Type" \
: disk = "Fujitsu-M2351 Eagle" : ctrlr = XY450 \
: a = 0, 15884 : b = 18, 33440 : c = 0, 772800 : g = 55, 722200

partition = "Fujitsu-M2351 Eagle" \
: disk = "Fujitsu-M2351 Eagle" : ctrlr = XD7053 \
: a = 0, 16560 : b = 18, 34040 : c = 0, 772800 : g = 55, 722200

partition = "Fujitsu-M2351 Eagle Old Type" \
: disk = "Fujitsu-M2351 Eagle" : ctrlr = XD7053 \
: a = 0, 15884 : b = 18, 33440 : c = 0, 772800 : g = 55, 722200

partition = "Fujitsu-M2333" \
: disk = "Fujitsu-M2333" : ctrlr = XD7053 \
: a = 0, 16080 : b = 24, 33500 : c = 0, 550070 : g = 74, 500490

partition = "Micropolis 1355" \
: disk = "Micropolis 1355" : ctrlr = MD21 \
: a = 0, 16048 : b = 59, 33456 : c = 0, 276896 : g = 182, 227392

partition = "Toshiba MK 156F" \
: disk = "Toshiba MK 156F" : ctrlr = MD21 \
: a = 0, 15980 : b = 47, 33660 : c = 0, 277100 : g = 146, 227460

```

10.6. format Command Reference

This section describes the commands available in `format`. The user interface is organized as a tree of menus. Each menu is explained separately in the following sections.

The Command Menu

The *command menu* is entered when the `format` program is invoked. It is identified by the `format>` prompt. The following commands are included in the *command menu*:

```

disk          - select a disk
type          - select (define) a disk type
partition     - select (define) a partition table
current       - describe the current disk
format        - format and analyze the disk
repair        - repair a defective sector
show          - translate a disk address
label         - write label to the disk
analyze       - surface analysis
defect        - defect list management
backup        - search for backup labels
quit

```

`disk` - select a disk

This command is used to select the *current disk*. The *current disk* is the disk that is currently being operated on by `format`. Most of the commands in `format` cannot be run unless a *current disk* is selected. The *current disk* can be set on the command line, so the `disk` command is not always necessary. When `disk` is run, it lists the boot lines for all the disks that were found during the disk search. The disks are numbered, and the *current disk* is selected by entering the number next to the desired disk. If there is already a *current disk* when `disk` is run, it is automatically the default choice. The *current disk* may be changed at will without worrying about harming the state of any of the disks. This allows you to work on several disks in one invocation of `format`. Below is an example of the `disk` command. At the time it was run, `xy0` was the *current disk*.

```
format> disk

AVAILABLE DISK SELECTIONS:
  0. xy0 at xyc0 slave 0
     xy0: <drive type unknown>
  1. xy1 at xyc0 slave 1
     xy1: <Fujitsu-M2333 cyl 821 alt 2 hd 10 sec 67>
  2. xd0 at xdc0 slave 0
     xd0: <drive type unknown>
  3. sd0 at si0 slave 0
     sd0: <drive type unknown>
Specify disk (enter its number) : 3
```

Note the messages displayed after the choice was made. If the disk type is known, it is printed inside the angle brackets ('<>'). If the disk type is not known,

```
<drive type unknown>
```

is printed. If the type is known, an attempt is made to see if the disk is formatted. The results of this test are shown in the square brackets ('[]'). If you know that the result of this test is incorrect, something is wrong. One possibility is that the disk type assigned to the disk is incorrect. Be sure that the type is correct, and fix it with the `type` command if it is not. If that isn't the problem, there is probably a hardware problem with your system. Exit `format` immediately and diagnose the problem. Continuing in `format` only causes damage to the disk. If the disk appears formatted, an attempt is made to read the defect list stored at the back of the disk. The result of this is also displayed in the square brackets. If a defect list was found, the *current defect list* will be set equal to it. If not, the *current defect list* is set to null. All disks that were originally formatted by `format` should have a defect list on them. Those that were formatted by `diag` may or may not have a defect list, depending on which version of `diag` was used. If the selected disk does not have a defect list, you can create one for it. See the section on [@TitleOf\(using format\)](#) for details.

`type` - select (define) a disk type

This command is used to specify the *current disk type*. The *current disk type* describes the physical characteristics of the *current disk*. Most of the commands in `format` cannot be run unless the *current disk type* is set. The *current disk type* can be set from the command line, and is set automatically if the *current disk* is labeled. Thus, the `type` command is not always necessary.

When `type` is run, it lists the disk types that are supported for the *current disk*. It also lists *other* as a choice in case you need to define your own disk type. The types are numbered, and the *current disk type* is selected by entering the number next to the desired disk type. If there is already a *current disk type* when `type` is run, it will automatically be the default choice.

Exercise extreme caution when using the `type` command. If the *current disk* is labeled correctly, the *current disk type* is set automatically when the disk is selected. Thus, you should never have to run the `type` command unless you have a non-Sun supplied disk or a disk that was incorrectly labeled. Specifying the wrong type for a disk can cause a variety of strange behaviors. In many cases, a slew of errors occurs during one of the other commands, because `format` has the wrong geometry for the disk. Another sign of the wrong disk type is if `format` claims the disk is unformatted when you know it's formatted. Because it may be hard to correct the situation by the time you discover it, you should always be certain of the correct disk type before selecting it.

Below is an example of the `type` command. At the time it was run, `sd0` was the *current disk*. Its type was unknown, and it was a non-Sun supplied disk.

```
format> type

AVAILABLE DRIVE TYPES:
  0. Micropolis 1355
  1. Toshiba MK 156F
  2. Micropolis 1558
  3. other
Specify disk type (enter its number) : 2
selecting sd0: <Micropolis 1558>
[disk unformatted, no defect list found]
```

`partition` - select (define) a partition table

This command is used to enter the *partition menu*. The *partition menu* contains commands to define a partition table and select the *current partition table*. The *current partition table* is used when labeling the *current disk*. More information on the `partition` command is given in a later section.

`current` - describe the current disk

This command displays a brief summary of the *current disk*. It allows you to see which disk is currently selected. The format of the summary is similar to the boot line for a disk. If *current disk* and *current disk type* are both set, the summary will show which disk is selected and describe its physical attributes. If either of these is not set, the summary will reflect that fact. Below are some examples of the `current` command. They show the various summaries that will be printed depending on whether *current disk* and *current disk type* are set.


```
format> current
No Current Disk.

format> current
Current Disk = xd0: <drive type unknown>

format> current
Current Disk = sd0: <Micropolis 1558 cyl 1218 alt 2 hd 15 sec 35>
```

format - format the disk

This command is used to format part or all of the *current disk*. Both *current disk* and *current disk type* must be set to run this command. Also, the *current defect list* must be initialized. If the disk is brand new, you must format the whole disk. Failure to do so will leave the disk in a very bad state. After that, the only time formatting is necessary is when the *current defect list* has been modified using commands in the defect menu. In this case, only the portion of the disk affected by the change needs to be reformatted. When `format` is run, it first asks for the bounds of the `format` operation, provided the disk controller supports such behavior. The default choices for the starting and ending blocks are always the first and last blocks on the *current disk*. Using the default values will cause the entire disk to be formatted. If only part of the disk needs to be formatted, there are some things to keep in mind. First, the starting and ending blocks you specify are both inclusive. This means that both of the sectors specified as the bounds will be included in the `format` operation. For example, if you are using a disk with 9 heads and 66 sectors per track and you want to format all of cylinder 10, you should enter:

```
format> format
Enter starting block number [0, 0/0/0]: 10/0/0
Enter ending block number [551409, 822/9/66]: 10/9/66
Ready to format, continue?
```

Note that the ending head and sector numbers specify the last block on the cylinder. You can tell this because they are equal to the default values, which always indicate the last block on the disk. A more convenient way to specify the same bounds would be to enter:

```
format> format
Enter starting block number [0, 0/0/0]: 10/
Enter ending block number [551409, 822/9/66]: 10/$/$
Ready to format, continue?
```

For a complete explanation on how to enter block numbers, see the section on *Interacting with format*.

Another concern when formatting only part of the *current disk* are the restrictions imposed by your disk controller. Some controllers can only format the entire disk, and others can format only whole tracks. If the bounds you enter do not

satisfy the requirements of the controller, an error message is printed and the command aborted.

After the bounds are specified, if `format` is being run interactively instead of from a command file, you are asked to verify that you want to do the `format` operation. Also, if the portion of the disk specified for formatting overlaps any mounted file systems, you are warned and given another chance to cancel the command. It is strongly recommended that you never go ahead with the `format` under these circumstances. Doing so could cause the system to crash since the mounted file system will be destroyed. This problem cannot occur when running from a command file, since the `disk` command will generate an error if a disk with a mounted file system is selected to be the *current disk*.

At this point, the `format` operation starts. While the `format` is in progress, you cannot interrupt the command with a `CTRL-C`. This is to avoid leaving the disk partially formatted. The specified portion of the disk is formatted, and all defects in the *current defect list* that fall into the formatted area are repaired. If something went wrong, a

```
format failed
```

message is displayed, and the command is aborted. If everything went well, a

```
format succeeded
```

message is displayed. At this point, `CTRL-C` interrupts are enabled again, so you can stop the command at any time without harming the disk. If the surface analysis parameters are not set up to verify after formatting, the command is finished. However, the default parameters, which are strongly recommended for peace of mind, have verification enabled.

Two passes of the write test will be run over the portion of the disk that was just formatted. Unless you have changed the analysis parameters, automatic repair is enabled during the analysis. This means that any defects found are repaired if possible.

```
format> format
Ready to format. Formatting cannot be interrupted
and takes 11 minutes (estimated). Continue? y
Beginning format. The current time is Wed Feb 10 18:31:57 1988

Verifying media...
    pass 0 - pattern = 0xc6dec6de
    1217/11/14

    pass 1 - pattern = 0x6db6db6d
    1217/11/14
Total of 0 defective blocks repaired.
```

`repair` - repair a defective sector

This command is used to repair a defective sector on the *current disk*. Both *current disk* and *current disk type* must be set to run this command. Also, the *current defect list* must be initialized, and the *current disk* must be formatted. The `repair` command is optionally supported by a disk controller, so some may not support repairing defective sectors. If you run `repair` on a controller of this type, it will print an error message and abort the command. In this case, you must use the `add` command to add the sector to the *current defect list* then reformat the *current disk*.

When `repair` is run, it first asks for the sector number to be repaired. This is the logical block number of the defective sector. It is typically the block number reported by a SunOS error message, or by a surface analysis command in `format`. After the block number is specified, if `repair` is run interactively instead of from a command file, you are asked to verify that you want to do the repair operation. Also, if the sector being repaired is within a mounted file system, you are warned and again given the chance to cancel the command. You should exercise caution repairing a sector in a mounted file system. If the sector is accessed by the system while the repair is occurring, the result is unpredictable. Thus, `repair` should be run only within a mounted file system if the system is idle.

At this point, the repair starts. While the repair is in progress, you cannot interrupt the command with a **CTRL-C**. This is necessary to insure the integrity of the *current defect list*. If something goes wrong, a

```
repair failed
```

message is displayed, and the command is aborted. If everything went well, a

```
repair succeeded
```

message is printed and the defective sector is automatically added to the *current defect list*. The operating system is notified of the new defect, so the repair takes effect immediately. Below are some examples of the `repair` command.

```
format> repair
Enter block number of defect: 73/0/23
Ready to repair defect, continue? y
Repairing block 35063 (73/0/23)...done.

format> repair
Enter block number of defect: 48997
Ready to repair defect, continue? y
Repairing block 48997 (73/1/20)...done.
```

`show` - show a disk address

This command is used to show a disk block in several of the formats understood by `format`. Because the disk geometry is necessary to translate the block number, both *current disk* and *current disk type* must be set to run this command. When `show` is run, you are prompted for a disk block number. The block number can be entered in any form understood by `format`. For a full

explanation of specifying block numbers, see the section on *Interacting with format*.

Once the block number is entered, it will be echoed in decimal, hexadecimal, and cylinder/head/sector format. The `show` command allows you to translate the decimal block numbers given in SunOS error messages into cylinder, head, and sector numbers. This may be useful in determining whether your disk is deteriorating in a pattern, such as a specific head going bad. This command in no way affects the *current disk*. It is intended for informational purposes only.

```
format> show
Enter a disk block: 28/9/34
Disk block = 19397 = 0x4bc5 = (28/9/34)
format> show
Enter a disk block: 12345
Disk block = 12345 = 0x3039 = (18/4/17)
format> show
Enter a disk block: 334466
Disk block = 334466 = 0x51a82 = (499/2/2)
format>
```

label - label the disk

This command is used to label the *current disk*. Both *current disk* and *current disk type* must be set to run this command. Also, the *current disk* must be formatted or `label` will fail. When `label` is run, it first checks to see if the *current partition table* is set. If set, the *current partition table* is used when the disk is labeled. If it's not set, `format` uses the first partition table on the list of known tables for the *current disk type*. When this occurs, you are informed that `format` will use the default table. If there are no known partition tables for the *current disk type*, an error message is printed and the command aborted.

At this point, if `format` is being run interactively, you are asked to verify if you want to label the *current disk*. Also, if there are any mounted file systems on the *current disk*, you are warned and again given the chance to cancel the command. You should exercise caution when labeling a disk with file systems on it. If the file systems are mounted, changing the partition information for the partition containing the file system may cause the system to crash. Even if the file system is not mounted, changing its partition information may destroy the files in it. You should always back up the system before relabeling disks with file systems.

After all checks have been completed, the primary and backup labels are written on the *current disk*. The kernel is also notified of the information in the new label, so any changes will take effect immediately. During labeling, **CTRL-C** interrupts are disabled, so the process cannot be stopped midway. This is necessary to prevent a disk from having only some copies of the label updated.

analyze - surface analysis

This command is used to enter the *analyze menu*. The *analyze menu* contains commands to analyze the surface of a disk for media defects. More information on the `analyze` command is given in a later section.

defect - defect list
management

This command is used to enter the *defect menu*. The *defect menu* contains commands to define and manipulate the *current defect list*. The *current defect list* is used when formatting and repairing the *current disk*. More information on the defect command is given in a later section.

backup - search for backup
labels

This command is used to restore the primary label from backup copies stored on the *current disk*. If the primary label was corrupted, `format` has no way of knowing the disk type or partition table for the disk. You will have to specify the disk type with the `type` command. However, the partition table for the *current disk* may be recovered by searching for backup labels. Since the location of the backup labels is geometry dependent, both *current disk* and *current disk type* must be set to run this command. Also, the *current disk* must be formatted or the command will fail.

When `backup` is run, it will first check to see if the primary label was present on the *current disk*. If it was, it informs you that the disk was labeled, and allows you to cancel the command. If the disk was not labeled, or you choose to go on anyway, an attempt is made to read each of the five backup labels stored on the *current disk*. If no good backup label is found, the command ends. If a good copy is found, it is checked against the geometry information of the *current disk* for correctness. If it passes these checks, the information in the label is used to set the *current partition table*. At this point, the primary label is rewritten, and the *current defect list* is copied to the disk. Below are some examples of the backup command.

```
format> backup
Disk had a primary label, still continue? y
Searching for backup labels...found.
Restoring primary label and defect list.
format>

format> backup
Searching for backup labels...not found.
format>
```

The fact that `backup` also rewrites the *current defect list* onto the disk is worth discussion. This feature is very useful if the *current disk* has had its defect list corrupted. It allows you to recreate the defect list, then use the `backup` command to write it back to the *current disk*. This will restore the defect list without requiring you to reformat the disk. However, a word of caution is in order. If the *current defect list* does not reflect the actual state of the media when `backup` is run, the defect list on the disk will become incorrect. Thus, extreme caution should be taken when you run `backup` after manipulating the defect list in any way.

The Partition Menu

The *partition menu* is entered by running the `partition` command. It is identified by the `partition>` prompt. The following commands are included in the *partition menu* :

```

a      - change 'a' partition
b      - change 'b' partition
c      - change 'c' partition
d      - change 'd' partition
e      - change 'e' partition
f      - change 'f' partition
g      - change 'g' partition
h      - change 'h' partition
select - select a predefined table
name   - name the current table
print  - display the current table
label  - write partition map and label to the disk
quit

```

a - change 'a' partition

This command is used to modify the *a* partition of the *current partition table*. If the *current partition table* is named when `a` is run, a new, unnamed partition table will be created with the changed *a* partition, and the *current partition table* will be set equal to it. If the *current partition table* is unnamed when `a` is run, the *a* partition in the current table is modified. This is explained more under the `name` command. When the `a` command is run, it first displays the current values for the *a* partition of the *current partition table*. It then asks you for new values for the starting cylinder and the number of blocks in the partition. The current values are always the default. Below is an example of the `a` command.

```

partition> a

      partition a - starting cyl      0, # blocks  ...

      Enter new starting cyl [0]: <cr>
      Enter new # blocks [16080, 24/0/0]: 25/
partition>

```

b - change 'b' partition

This command is used to modify the *b* partition of the *current partition table*. The interface to the `b` command is identical to that of the `a` command.

c - change 'c' partition

This command is used to modify the *c* partition of the *current partition table*. The interface to the `c` command is identical to that of the `a` command.

d - change 'd' partition

This command is used to modify the *d* partition of the *current partition table*. The interface to the `d` command is identical to that of the `a` command.

- e - change 'e' partition This command is used to modify the e partition of the *current partition table*. The interface to the e command is identical to that of the a command.
- f - change 'f' partition This command is used to modify the f partition of the *current partition table*. The interface to the f command is identical to that of the a command.
- g - change 'g' partition This command is used to modify the g partition of the *current partition table*. The interface to the g command is identical to that of the a command.
- h - change 'h' partition This command is used to modify the h partition of the *current partition table*. The interface to the h command is identical to that of the a command.

select - select a predefined table

This command is used to set the *current partition table* equal to a predefined partition table. First, we need an explanation of what a predefined table really is.

Whenever a *current disk* that is labelled is selected, the *current partition table* will automatically be set to partition information in the label. If this information does not match a predefined table, then a new predefined table is created for it. The name of this new table is always "original xxN", where xx is the disk name and N the unit number. For instance, if xy0 was selected as the *current disk*, and it's partition information did not match a predefined table, the *current partition table* would be set to a new table called "original xy0".

Predefined tables can also be specified in the *format* data file. The default data file contains predefined partition tables for all Sun supported disks. You can add your own partition tables for other disks, or define new tables for Sun supported disks, simply by editing the data file. See *The format.dat File* for more details.

You can also create your own predefined partition tables by using other commands in the *partition menu*. The a-h commands allow you to modify the *current partition table*, then the name command allows you to freeze a copy of the *current partition table* and give it a name. Remember though, that defining a partition table this way only lasts for one invocation of *format*, whereas adding it to the data file causes it to be defined every time you run *format*.

Thus, when the *select* command is executed, there may be many different predefined partition tables for the *current disk*. They will be displayed in a list, and you can choose whichever one you want the *current partition table* set to. The default choice is always the current value of the *current partition table*. Below is an example of the *select* command.

```
partition> select
  0. Fujitsu-M2333
  1. original xy0
Specify table (enter its number) [1]: 0
partition>
```

name - name the current table

This command is used to name the *current partition table*. When a partition table gets named, it freezes the values of the table at their current settings. This way you can create your own predefined partition tables for use on multiple disks. You only need to name the *current partition table* if you have changed some of the partition information in it. Whenever you modify the partition information and the *current partition table* was named, a new, unnamed partition table is created to hold the new information. Once you are done modifying all the partitions you wish to change, you can name the resulting table so it can be used again and again. When you run the name command, it simply asks you for the name you wish to assign to the *current partition table*. If you want any spaces to appear in the name, you must remember to enclose the name in double quotes. Otherwise, only the first word you specify is used as the name. Below is an example of the name command. It shows how the *current partition table* is named after the name command is run.

```
partition> select
    0. Fujitsu-M2333
    1. original xy0
Specify table (enter its number) [1]: 0

partition> name
Enter table name (remember quotes): "new table"

partition> select
    0. Fujitsu-M2333
    1. original xy0
    2. new table
Specify table (enter its number) [0]: 0
```

print - display the current table

This command is used to print out the *current partition table*. It prints the name of the table (or "unnamed") and the values of all the partitions in the table. Below is an example of the print command.

```
partition> print
Current partition table (unnamed):
partition a - starting cyl      0, # blocks      16750 (25/0/0)
partition b - starting cyl     24, # blocks     33500 (50/0/0)
partition c - starting cyl      0, # blocks     550070 (821/0/0)
partition d - starting cyl     74, # blocks     20345 (30/3/44)
partition e - starting cyl      0, # blocks        0 (0/0/0)
partition f - starting cyl      0, # blocks        0 (0/0/0)
partition g - starting cyl    105, # blocks    480390 (717/0/0)
partition h - starting cyl      0, # blocks        0 (0/0/0)
partition>
```


The Analyze Menu

The *analyze menu* is entered by running the `analyze` command. It is identified by the `analyze>` prompt. The following commands are included in the *analyze menu*:

```
read    - read only test      (doesn't harm SunOS)
refresh - read then write     (doesn't harm data)
test    - pattern testing     (doesn't harm data)
write   - write then read     (corrupts data)
compare - write, read, compare (corrupts data)
purge   - write, read, write  (corrupts data)
print   - display data buffer
setup   - set analysis parameters
config  - show analysis parameters
quit
```

`read` - read only test (doesn't harm the system)

This command is used to analyze the surface of the *current disk* for media defects. Both the *current disk* and *current disk type* must be set to run this command. Also, if the analysis parameters are set up to do automatic repairing, the *current defect list* must be initialized. The `read` command is the one type of surface analysis that can be safely run on mounted file systems. It simply reads the sectors specified, looking for media related errors.

When the `read` command is run, it uses the analysis parameters to control the operation of the command. See the description of the `setup` command for a complete explanation of the analysis parameters and what they do. If `read` is being run interactively, it asks you to verify that you want to do the analysis operation. Also, if the boundaries of the analysis overlap with any mounted file systems, you are warned and again given the chance to cancel the command. In the case of the `read` command, there is no danger from running on a mounted file system, unless automatic repairing is enabled. You should only allow automatic repairing on a mounted file system if the system is idle, so a repair does not interfere with the operation of the system. You can easily monitor the progress of the command since it is continuously displayed. Below is an example of the `read` command.

```
analyze> read
Ready to analyze (won't harm the system). This takes a long time,
but is interruptible with CTRL-C. Continue? y
      pass 0
      pass 1
6/6/50
Total of 0 defective blocks repaired.
analyze>
```

If automatic repairing is enabled, then all media defects found during the analysis will be repaired. Any time an error is detected in a command, the SunOS driver will notify `format` whether the error indicates a media defect or not. If it does indicate a media defect, then the area around the error is analyzed carefully. If the defective sector can be pinpointed, it is automatically repaired.

You do not need to let the `read` command run to completion. Any time you are satisfied that the disk is sufficiently tested, you can abort the command with a **CTRL-C**.

`refresh` - read then write
(doesn't harm data)

This command is very similar to the `read` command, but performs a more comprehensive check of the media. It will not change the data on the *current disk*, but should not be run on a mounted file system. If the warning concerning mounted file systems is displayed, you should never proceed. The `refresh` command reads each sector of the specified area, then writes the same data back to the disk. While it does this, it is looking for media related errors. In all other respects, it is identical to the `read` command.

`test` - pattern testing (doesn't harm data)

This command is very similar to the `read` command, but performs a very comprehensive check of the media without corrupting the data on the *current disk*. For each sector in the analysis region, it reads the data and stores it away. Test then writes and reads a specific data pattern to the disk. Finally, the original data is put back on the disk. Although this results in no change of the disk data, it should not be run on a mounted file system. If the warning concerning mounted file systems is displayed, you should never proceed. Below is an example of the `test` command.

```
analyze> test
Ready to analyze (won't harm data). This takes a long time,
but is interruptable with CTRL-C. Continue? y
    pass 0 - pattern = 0xc6dec6de
    pass 1 - pattern = 0x6db6db6d
    pass 2 - pattern = 0x0
    pass 3 - pattern = 0xffffffff
    pass 4 - pattern = 0xaaaaaaaa
73/9/47
Total of 0 defective blocks repaired.
analyze>
```

`write` - write then read
(corrupts data)

This command is the quickest way to do pattern testing on the *current disk*. The interface to the `write` command is identical to that for the other analysis commands. However, the `write` command does not preserve the data on the *current disk*. This is the type of analysis that is run automatically by the `format` command after it has formatted the *current disk*. A series of patterns are written to the disk then read back. While this is happening, `format` is looking for media related errors. Because this command does not preserve the data on the *current disk*, it should never be run on a portion of a disk that contains valuable data or a mounted file system.

`compare` - write, read,
compare (corrupts data)

This command is the most comprehensive way to check the *current disk* for media defects. It is similar to the `write` command, but it also compares the data pattern read back from the disk to the one originally written. Any discrepancy in the data pattern is shown as an error message. Like the `write` command, `compare` should never be run on a portion of the *current disk* that

contains valuable data or a mounted file system.

`purge -`
`write,read,compare,write`
 (corrupts data)

This command sanitizes the disk. All data is removed from the disk by multiple pattern writes to ensure that it can't be retrieved by any means. Data is removed by writing three distinct patterns over the entire disk (or section of the disk), then writing an alphanumeric pattern if the verification passes. Make sure to save the defect list to tape or another disk before executing this command. To read a block after the command is done, use the read only test on a single block and then the print command.

`print - display data buffer`

This command is used to display the contents of the format data buffer. This is not normally necessary for disk maintenance. This command is only useful if you need to look at the data in a certain sector of the *current disk*. Using the `read` command to fill the buffer with the data desired, the `print` command can then be used to get a hexadecimal dump of the data. If you don't want to view the entire data buffer, you can use a `[CTRL-C]` to abort the command at any time.

`setup - set analysis parameters`

This command is used to set the parameters for the actual surface analysis commands. This allows you to specify the parameters once, then run analysis multiple times without reinitializing the parameters. This is the list of information queried for in the `setup` command, along with the initial value of each item:

- boundaries of the analysis	(entire disk)
- number of analysis passes	(2)
- enable/disable automatic repair	(enabled)
- stop/continue after an error	(continue)
- random/built-in data patterns	(built-in)
- size of each analysis transfer	(126 sectors)
- enable/disable post-format analysis	(enabled)
- enable/disable extended messages	(disabled)
- restore defect list	(enabled)
- restore disk label	(enabled)

The exact format of the questions asked varies slightly depending on the current state of the parameters. However, every question has the current value as the default, so entering just a carriage return will leave that parameter unchanged.

The boundaries of the analysis are specified in two ways. You can either specify the entire disk, or you can specify the beginning and ending block numbers. If you use block numbers to specify the bounds, the block numbers are inclusive. Specifying the entire disk will not hurt the label and defect information stored on the disk. No matter which surface analysis command you use, that information is carefully saved and restored when the command is completed.

The number of passes to perform can also be specified in two ways. You can either tell `format` to loop continuously, or you can give a specific number of passes to run. In loop mode, any surface analysis command will run until you stop it with a `[CTRL-C]`.

Automatic repair can be enabled or disabled, depending on how much control you want over the analysis procedure. When it's enabled, any sector that can be identified as causing a media related error will be repaired immediately. This is fine for most cases. However, you can disable this feature if you want discretionary control over what gets repaired. When automatic repairing is disabled, it simply notifies you of all the defective sectors that were found. Also, some controllers may not support repairing. If this is the case, you will simply be notified of all the defective sectors found, independently of whether automatic repair is enabled or disabled.

You can choose to stop when the first error is encountered, or continue in spite of the errors. Normally, analysis is used to find any and all errors, so continuing is the logical choice. However, there might be circumstances which cause you to want to stop as soon as any error is encountered.

The data patterns used for the pattern testing commands come from one of two sources. You can select random data patterns, which cause a different random number to be generated for each pass, or you can choose the built-in data patterns, which are designed to maximize the chance of finding a pattern sensitive failure.

The size of each analysis transfer is fully programmable. Any size can be specified between one sector and 126 sectors. For normal surface analysis, scanning 126 sectors at a time is most efficient. It reduces the total time taken by the analysis commands, but still catches any errors. However, if you are trying to pinpoint an elusive error, you may want to scan a smaller number of sectors.

By default, surface analysis is run over the newly formatted portion of the *current disk* whenever the `format` command is executed. However, you are free to disable this feature if you want to `format` without running analysis.

Here is an examples of the `setup` command.

```
analyze> setup
Analyze entire disk [yes]? n
Enter starting block number [0, 0/0/0]: <cr>
Enter ending block number [551409, 822/9/66]: 20/$/$
Loop continuously [no]? y
Repair defective blocks [yes]? n
Stop after first error [no]? <cr>
Use random bit patterns [no]? <cr>
Enter number of blocks per transfer [126, 0/1/59]: <cr>
Verify media after formatting [yes]? <cr>
Enable extended messages [no]? <cr>
Restore defect list [yes]? <cr>
Restore disk label [yes]? <cr>
```

`config - show analysis
parameters`

This command is used to display the current settings for the surface analysis parameters. It prints out the same lines as the `setup` command, but fills in the

fields for you with the current value of each parameter. Below is an example of the config command.

```
analyze> config
  Analyze entire disk? no
  Enter starting block number: 0, 0/0/0
  Enter ending block number: 14069, 20/9/66
  Loop continuously? yes
  Repair defective blocks? no
  Stop after first error? no
  Use random bit patterns? no
  Enter number of blocks per transfer: 126, 0/1/59
  Verify media after formatting? yes
  Enable extended messages? no
  Restore defect list? yes
  Restore disk label? yes
analyze>
```

The Defect Menu

The *defect menu* is entered by running the defect command. It is identified by the defect> prompt. The following commands are included in the *defect menu*:

```
restore    - set working list = current list
original   - extract manufacturer's list from disk
extract    - extract working list from disk
add        - add defects to working list
delete     - delete a defect from working list
print      - display working list
dump       - dump working list to file
load       - load working list from file
commit     - set current list = working list
create     - recreates manufacturers defect list on disk
quit
```

restore - set working list =
current list

This command is used to set the *working defect list* equal to the *current defect list*. This command is useful if you change the *working defect list* then decide that you want to undo the changes. As long as you have not committed the changes, running *restore* will successfully undo them. If the *current defect list* is null when *restore* is run, it will set the working defect to null also. When the *restore* command is run, it asks you to verify that you want to do the restore. It then reinitializes the *working defect list*. Below is an example of the *restore* command.

```
defect> restore
ready to update working list, continue? y
working list updated, total of 25 defects.
defect>
```

`original - extract
manufacturer's list from disk`

This command is used to create a *working defect list* equal to the manufacturer's defect list for the *current disk*. This is usually used when a disk is brand new (never formatted), and you wish to create the original defect list for the disk. All disks shipped from Sun are formatted at the factory, so this command is not necessary. This command is also useful if you repair several defects on the disk then discover it was some problem other than the disk media. By going back to the original defect list and reformatting the drive, you can remove all the erroneous repairs.

The `original` command is optionally supported by a disk controller, so some may not allow you to extract the manufacturer's defect list this way. Some controllers can extract the manufacturer's list if the disk is unformatted, but cannot do it once the disk has been formatted. If you can't use the `original` command, you will have to enter the defects from your hard copy of the defect list using the `add` command. In either case, you must use the `commit` command and then reformat the *current disk* for the new defects to be used.

The `original` command is not interruptable by a `CTRL-C`. This is necessary so that the *working defect list* does not get partially updated. Extracting the manufacturer's defect list can take a long time on some disks, so be ready to wait for a while. When `original` is run, it notifies you that the command is not interruptable, and asks for verification that you really want to do the command. It then updates the *working defect list* with the manufacturer's list. Below is an example of the `original` command.

```
defect> original
Ready to update working list. This cannot be interrupted
and may take a long while. Continue? y
Extracting manufacturer's defect list.
Extraction complete.
Working list updated, total of 25 defects.
defect>
```

`extract - extract working list
from disk`

This command is used to extract the current defect list from the *current disk* media. This is not `format's current defect list`, it is the list of defects that reflects the actual state of the *current disk*. This list is put into `format's working defect list`. The `commit` command can then be used to set `format's current defect list`, so the defects can be used by other commands.

This command is only necessary if the copy of the defect list stored on the *current disk* by `format` becomes corrupted. The other reason to use this command is to create a defect list for a disk that is older, and therefore never had a defect list on it. This should be done to all your older disks whenever convenient, so that all your disks have defect lists on them.

The `extract` command is optionally supported by a disk controller, so some may not allow you to extract the disk's current defect list this way. Some controllers can extract the current defect list once the disk has been formatted, but not when the disk is brand new. This only affects you if you did not get your disks from Sun, since all Sun disks are formatted at the factory. If the `extract`

command cannot be used, you can use a copy of the defect list saved with the `dump` command to initialize the *working defect list* instead. This is why you should always use the `dump` command to save the most up to date defect list you have for each disk. If you don't have a saved copy of the defect list, you will have to enter the defects by hand using the `add` command. In either case, you must use the `commit` command then reformat the *current disk* for the defects to be used.

The `extract` command is not interruptable by a `CTRL-C`. This is necessary so that the *working defect list* does not get partially updated. Extracting the current defect list can take a long time on some disks, so be ready to wait for a while. When `extract` is run, it notifies you that the command is not interruptable, and asks for verification that you really want to do the extraction. It then updates the *working defect list* with the defects it extracts from the disk media. Below is an example of the `extract` command.

```
defect> extract
Ready to update working list. This cannot be interrupted
and may take a long while. Continue? y
Extracting current defect list.
Extraction complete.
Working list updated, total of 25 defects.
defect>
```

`add` - add defects to working list

This command is used to add defects to the *working defect list*. This may be necessary for a variety of reasons. If the defect list stored on the *current disk* was corrupted or never existed, and you cannot extract the defect list, you can create the defect list by hand. Also, if you wish to repair a sector but the controller doesn't support repairing, you can add the defective sector to the *working defect list*, commit the defects to the *current defect list*, then reformat the *current disk*. This will have the effect of repairing the defective sector.

When the `add` command is run, it first asks you which mode the defects to be entered are in. The two modes are bytes from index and logical block. If the defects are from the manufacturer's defect list, they will be in bytes from index format. If you are adding a defective sector to the list on anything but a SCSI disk, it will be in logical block format. If you need to add defects of each type to the working defect list, you will have to run the command twice.

If you specify that the defects are in bytes from index format, you are then prompted for the defect's cylinder, head, bytes from index and length. The default value for the length is always -1, which means that the length isn't known. Some manufacturer's defect lists include the length of the defect, so you can specify the correct value if you have it. If you specify that the defects are in logical block format, you are instead prompted for the logical block number of the defective sector. This is the same value that you would specify to `repair` to fix the defective sector.

After the information about the defect has been entered, a summary of the defect is printed, along with which defect number it will become when it is added to the

working defect list. You are then asked to confirm that the defect should be added. If an affirmative is given, the defect is added to the *working defect list* and the next defect is prompted for. When you are finished adding defects to the *working defect list*, simply use **CTRL-C** to exit the loop. Below are some examples of the add command.

```
defect> add
    0. bytes-from-index
    1. logical block
Select input format (enter its number) [0]: 0

Enter Control-C to terminate.
Enter defect's cylinder number: 127
Enter defect's head number: 0
Enter defect's bytes-from-index: 2345
Enter defect's length (in bits) [-1]: 23
  num    cyl    hd    bfi    len    sec
   1     127     0   2345     23
defect number 1 added.

Enter defect's cylinder number: 400
Enter defect's head number: 5
Enter defect's bytes-from-index: 4567
Enter defect's length (in bits) [-1]:
  num    cyl    hd    bfi    len    sec
   2     400     5   4567
defect number 2 added.

Enter defect's cylinder number: ^C
defect>

defect> add
    0. bytes-from-index
    1. logical block
Select input format (enter its number) [0]: 1

Enter Control-C to terminate.
Enter defective block number: 34/5/20
  num    cyl    hd    bfi    len    sec
   3     34     5     5     20
defect number 3 added.

Enter defective block number: 500/0/0
  num    cyl    hd    bfi    len    sec
   4     500     0     0     0
defect number 4 added.

Enter defective block number: ^C
defect>
```


`delete` - delete a defect from working list

This command is used to delete defects from the *working defect list*. This may be necessary if you have added defects or repaired sectors then discovered that the media was not the problem. By deleting the defects from the *working defect list*, then committing to the *current defect list*, then reformatting the *current disk*, you can effectively erase the existence of the defects.

When the `delete` command is run, it asks you for the number of the defect that should be deleted. The defects are numbered in the `print` command, so you can get the number you need by running `print` and finding the defect you want to remove. When you enter the number, `format` then prints a summary of the defect and asks you to verify that you want to delete it. If an affirmative is given, the defect is deleted from the *working defect list*. A word of caution is warranted here. If you are planning on deleting more than one defect, you should do a `print` command after each `delete` and find the next defect you wish to remove. Once you delete a defect, all remaining defects in the list are renumbered, so other defects you may want to delete can change number on you. Thus, you should always look up each defect right before deleting it, and examine the summary printed carefully before confirming the delete. Below are some examples of the `delete` command.

```
defect> print
num    cyl    hd    bfi    len    sec
  1    127    0    2345    23
  2    400    5    4567
  3     34    5
  4    500    0
total of 4 defects.

defect> delete
Specify defect to be deleted (enter its number): 2
num    cyl    hd    bfi    len    sec
  2    400    5    4567
defect number 2 deleted.

defect> delete
Specify defect to be deleted (enter its number): 1
num    cyl    hd    bfi    len    sec
  1    127    0    2345    23
defect number 1 deleted.

defect> print
num    cyl    hd    bfi    len    sec
  1     34    5
  2    500    0
total of 2 defects.
defect>
```

`print` - display working list

This command is used to print out the *working defect list*. Remember that this is not necessarily the same as the *current defect list*, which is the list used when you

format the *current disk*. See the descriptions of the `restore` and `commit` commands for more details. The *working defect list* is displayed as a numbered list of defects. The number next to each defect can be used to identify the defect for other commands. The information shown for each defect is cylinder, head, bytes from index, length, and logical sector. Some of the information may not be known, so some of the fields may be blank for certain defects. Typically, if the defect was from the manufacturer's defect list, the sector number will not be known. If the defect was added as the result of surface analysis, the bytes from index will generally not be known. This is ok, since only one of these two pieces of information is necessary to locate the defect on the *current disk*.

The defects are shown in the order they are stored in the *working defect list*. This ordering allows `format` to consistently locate the defects on the *current disk*. All defects that are specified in bytes from index format are sorted so they appear in increasing order of their locations on the disk. All the defects specified in logical sector format are added to the end of the defect list in the order they are specified.

When the `print` command is run, it will display the *working defect list* one page at a time. You can use a carriage return `<cr>` to scroll to the next page. Also, you can abort the command at any time with a `CTRL-C`. Below is an example of the `print` command.

```

defect> print
num      cyl      hd      bfi      len      sec
  1         86         7    32641         2
  2        171         9    2352         2
  3        263         9    22757         3
  4        264         8     5853         6
  5        329         9    29195         3
  6        334         7      962         2
  7        350         1   39377         2
  8        352         8   27660         3
  9        377         9   21473         2
 10        383         2   13527         3
 11        403         1   31726         2
 12        486         1    7965         2
 13        513         7      260         2
 14        530         2   16558         3
 15        540         1    4325         3
 16        544         5   12517         3
 17        565         3   38136         6
 18        606         7   21997         3
 19        650         2   11262         3
 20        689         3   31911         2
 21        703         1   20294         3
 22        731         4   38475         3
- hit return for more -
 23        738         0   36468         3
 24         73         0                23
 25         73         1                20
total of 25 defects.
defect>

```

`dump` - dump working list to file

This command is used to dump the *working defect list* to a SunOS file for later retrieval with the `load` command. This command allows you to save an online copy of the *current disk's* defect list. By doing this, you insure that the defect list will not be lost, even if the data on the disk is corrupted.

If you are running `format` direct from the installation media you must take the extra step of saving the defect list to non-volatile storage, such as tape. Saving the defect list to tape is not absolutely required—you can also bring up `format` after you have booted the operating system. The extra step of saving the list to tape is recommended for peace of mind, however. Thus, if you format a new disk during `SunInstall`, you should run `format` again after you have booted the system and run the `dump` command to save the defect list. When you run the `dump` command, it asks you for the name of the file it should dump to. When you specify the defect file, it is a good idea to include the serial number of the disk in the name, so you can later match the file with the correct disk. The file is in `ascii`, so you can look at it. However, it contains a checksum, so you cannot edit the file by hand and have it still work in the `format` program. Also, since `dump` writes the *working defect list*, you should make sure that the *working defect list* and *current defect list* are the same before running the command.

Below is an example of the dump command.

```
defect> dump
Enter name of defect file: /usr/defects/2333_deflist.14257
defect file updated, total of 25 defects.
defect>
```

Note that a separate directory is used for defects for easy access and maintainability. In addition, the defect list name contains the drive's model and serial number (2333 and 14257, respectively). This is strongly advised.

load - load working list from file

This command is used to load a defect file created by dump into the *working defect list*. The file must have been created by the dump command, or load will not accept it. When the load command is run, it asks you for the name of the defect file. It then initializes the *working defect list* from the file, and reports how many defects were in the list. Remember that you still have to run the commit command to set the *current defect list*.

When running under MUNIX you will have to get the defect list from tape that you saved. You can then use the load command as shown. Below is an example of the load command.

```
defect> load
Enter name of defect file: /usr/defects/2333_defectlist.14257
ready to update working list, continue? y
working list updated, total of 25 defects.
defect>
```

commit - set current list = working list

This command is used to set the *current defect list* equal to the *working defect list*. When a *current disk* is selected, an attempt is made to read the defect list off the disk. If a list is found, both the *current defect list* and *working defect list* are set equal to it. If no list is found on the disk, then both are set to null. Several commands require that the *current defect list* be initialized before they will execute. If the *current disk* didn't have a list on it, then the commit command is the only way to initialize the *current defect list*. When you run the commit command, the *current defect list* always becomes initialized. If the *working defect list* was null, both lists are set to zero length instead. This is different than a null list, it means that the disk is assumed defect free. This may be useful if you have an older disk with no defect list and don't want to bother creating one.

If the *working defect list* was modified using any of the other commands in the defect menu, commit must be run so the *current defect list* reflects these changes. Also, the disk must be reformatted, so that the media itself uses the new defect list. If you want to (and your disk controller supports such behavior), you can only reformat the portion of the *current disk* that the defect list changes affect. However, you should be careful to include all necessary portions of the *current disk*, or some of your changes won't take effect.

If the repair command is used to add a sector to the defect list, the *current defect list* is updated automatically and no further action is necessary. Thus, the

commands in this menu should only be necessary if the controller doesn't support repairing or the defect list on the *current disk* was corrupted somehow.

When the `commit` command is run, it asks you to verify that you want to do the commit. It then tells you how many defects the *current defect list* now contains. If you have modified the *working defect list* and wish to use those modifications, you must run the `commit` command before reformatting the *current disk*. If you have modified the *working defect list* and you exit the *defect menu* without running the `commit` command, a warning message is printed telling you that there are uncommitted changes. You can then go back into the *defect menu* and run `commit` if desired. Below is an example of the `commit` command.

```
defect> commit
ready to update Current Defect List, continue? y
Current Defect List updated, total of 25 defects.
Disk must be reformatted for changes to take effect.
defect>
```

`create` - recreates manufacturers defect list on disk

This command is used to make a disk appear as if it just arrived from the disk manufacturer. This command is currently only supported on the SMD-4 disk controller. The defect entries in the current defect list are used. All of the data on the disk is destroyed when this command is executed. This command should only be used to allow the defect list to be readable when the disk is used on a controller that couldn't read the current format.

10.7. `format` Error Messages

This section lists the error messages that you may encounter while running `format`. For both SMD and SCSI drives, the messages are described in detail, and suggestions for how to get around the error are given.

Errors for both SMD and SCSI Drives

Backup Label Claims Different Type.

This error occurs when you are executing the `backup` command. It means that a backup label was found, but the disk type claimed by the backup label is not the disk type currently specified for the disk. This generally means that you selected the wrong disk type with the `type` command. Try running `type` again, this time specifying a different disk type. If you are sure that you are specifying the correct disk type, then the backup label must be incorrect. You should ignore it and build your own label, using `label` to write it on the disk.

Bad Block Map Table Overflow.

This error occurs when you are repairing a defective sector on an SMD disk. This can happen from `repair`, `format`, or any of the surface analysis commands. It means that the bad block table is full so another mapping cannot be added. This happens when the *current disk* already has 126 mapped sectors. This will generally happen only if the *current disk* is not configured for slip sectoring. You should configure all of your SMD disks for slip sectoring, since it has many benefits. Another common cause of this problem is running surface analysis with a bad cable. This will cause defects to be found that aren't really there, and result in many unnecessary mappings. You can remedy this situation by using the

`delete` command to remove the unreal defects from the *current defect list* then reformatting the disk with the `format` command. If a disk with slip sectoring has 126 genuinely mapped sectors, it is not within the Sun specification, and should be returned.

Bad keyword `<input>` -- line *N* of data file.

This error occurs during the startup processing of `format`, when it is reading in the data file. It means that `format` was expecting a keyword next in the file, but found something else. The line number of the error is printed so the problem can be easily located. The syntax of the data file consists of keywords followed by operators and values. The keywords recognized at any given time depend on what declaration is currently active. For a complete explanation of the data file syntax and a list of the keywords, see the *The format.dat File* section.

Can't open selected disk `<input>`.

This error can occur whenever the *current disk* or *current disk type* is changed. It means that `format` was unable to open the file representing the raw C partition of the *current disk*. This error shouldn't occur, since the disk must have been successfully opened during the startup processing or it would not have appeared as a choice. Exit the program and check the file carefully to see if something has happened to it.

Can't repair sector that is already mapped.

This error occurs when you are repairing a defective sector on an SMD disk. This can happen from `repair`, `format`, or any of the surface analysis commands. It means that the sector being repaired is already mapped to an alternate sector. A common cause of this problem is running surface analysis with a bad cable. This will cause defects to be found that aren't really there, and results in many unnecessary mappings. You can remedy this situation by using the `delete` command to remove the unreal defects from the *current defect list* then reformatting the disk with the `format` command. If the sector being repaired is really defective, there is not much you can do. If the *current disk* is configured for slip sectoring, you can try running surface analysis on the alternate cylinders. If the defect in the alternate sector shows up during this analysis, it will be repaired by slipping the sector. This will cause the problem to go away. If this doesn't work, the only thing you can try is reformatting the alternate cylinders in hopes that the defect won't show up again.

Controller does not support extracting current defect list.

The `extract` command is optionally supported by various controller types. If the controller for the *current disk* doesn't support the command, an attempt to run `extract` will generate this message. If the *current defect list* cannot be initialized with `extract`, you can enter the defects manually with the `add` command.

Controller does not support extracting manufacturer's defect list.

The `original` command is optionally supported by various controller types. If the controller for the *current disk* doesn't support the command, an attempt to run `original` will generate this message. If you want to initialize the *current defect list*, you can enter the defects manually with the `add` command.

Controller does not support repairing.

The ability to repair defective sectors is optionally supported by various controller types. If the controller for the *current disk* doesn't support repairing, any attempt to repair a sector will generate this message. This can happen from

`format`, `repair`, or any of the surface analysis commands. You can repair the defective sector by using `add` to add it to the *current defect list* then reformatting that portion of the *current disk*.

Controller firmware is outdated and doesn't support extraction.

This error occurs when you are running the `extract` command to extract the manufacturer's defect list and the controller firmware has not been upgraded to handle this. The only controller that causes this message to be generated is the Xylogics 450, which formerly did not support extraction. To solve the immediate problem, you can create the defect list manually with the `add` command. However, you should get a firmware update from the manufacturer anyway, since running with outdated firmware can have unfortunate consequences.

Controller requires formatting of entire tracks.

This message can occur when executing the `format` command. It means that the controller does not support formatting of a portion of a track. You will have to respecify the bounds of the format operation in terms of whole tracks.

Controller requires formatting the entire drive.

This message can occur when executing the `format` command. It means that the controller does not support the option of formatting a portion of the disk. You must run `format` over the entire disk. Remember to back up any files on the disk first.

Current Defect List must be initialized to do automatic repair.

This error can occur when executing any of the surface analysis commands. All of these commands require a *current defect list* to be initialized if automatic repairing is enabled. The *current defect list* can be initialized with the `commit` command.

Current Defect List must be initialized.

This warning can occur when you execute the `format` or `repair` commands. It means that you need to initialize a *current defect list* in order to proceed, since both of the above commands require it. You can initialize the *current defect list* with the `commit` command.

Current Disk Type is not set.

This error can occur when executing many of the commands in the *command menu* without first having set the *current disk type*. This parameter must be set for many of the commands to run. If this error occurs after you have already selected a *current disk* with the `disk` command, you need to specify the disk type with the `type` command. Otherwise, the type is automatically set when you select a *current disk*.

Warning: Current Disk has mounted partitions.

This warning occurs when you run the `disk` command to select a disk that has mounted partitions. You should exercise caution when working on a disk with mounted partitions. If possible, you should unmount the partitions before proceeding.

Current Disk is not set.

This error can occur on any of the commands which require the *current disk* parameter to be set (e.g. the `format` command). You can select the *current disk* by running the `disk` command. After you do so, your selection remains the *current disk* until you change it with another `disk` command.

- Current Disk is unformatted. This message is displayed when you try to execute a command that only makes sense if the *current disk* is formatted. Examples of such commands are the `repair`, `label`, and `backup` commands, which all assume a formatted disk. If you know the current disk is formatted, then something is wrong with your system. Exit `format` immediately, then use diagnostics to locate the problem.
- Current Partition Table is not set, using default. This message can be printed when you run the `label` command. It means that the *current partition table* is not initialized so `format` is attempting to use the default table specified in the data file. If there is no default, `label` will print out a message to that effect and give up. If you do not want to use the default, initialize the *current partition table* with the *partition menu* commands.
- Data miscompare error (expecting 0xN, got 0xM) at X/Y/Z, offset = 0xQ. This error can occur when running the `compare` command. It means that `format` detected a mismatch between an expected data pattern and the actual data pattern on a sector of the *current disk*. `format` then prints out the defective block number and attempts to repair it if automatic repairing is enabled.
- Defect file is corrupted, working list set to NULL. This error occurs when executing the `load` command. It means the specified defect file was found to have inconsistencies such as an improper number of parameters in a defect entry or a bad checksum on the defect file as a whole. `format` therefore assumes that the *working defect list* contains garbage and reinitializes it to NULL. You will have to rebuild the working defect list manually with the `add` command.
- Defect file is corrupted. This error occurs when executing the `load` command. It means that the header of the specified defect file was found to be logically inconsistent. The working defect list is not updated when this error occurs. Check to be sure you specified the defect file correctly. If so, you will have to rebuild the working defect list manually with the `add` command.
- Defect file <input> is not accessible. This error occurs when executing the `load` command. It means either that you do not have the proper permissions to access the specified defect file or that the defect file doesn't exist. Check the defect file and its access permissions and retry the command.
- Disk is not fully encoded with defect info. This error occurs when you execute the `original` command on an SMD disk. It means that multiple failures have occurred in extracting the manufacturer's defect list from the disk. Although you have the option of continuing, it is recommended that you exit and create the defect list by hand with the `add` command using the printed manufacturer's defect list.
- Disk is not fully formatted. This error occurs when executing the `extract` command. It means that the disk is in a partially formatted state. If you know the current disk is fully formatted, there is something wrong with your system. You should exit `format` immediately and use diagnostics to locate the problem. It is possible to continue the extraction after this error occurs, but it is not recommended.

Error: found disk attached to unsupported controller type <i>N</i> .	This error occurs when <code>format</code> cannot identify the controller attached to one of the disks. This should never happen on a Sun supported controller.
Error: unable to malloc more space.	This error can occur any time there is not enough memory to satisfy a request by <code>format</code> for temporary storage. If you are running <code>format</code> online, try to re-run it when there is less activity on the system.
Error: unexpected ioctl error from <code>xd</code> driver.	This error indicates an internal inconsistency in <code>format</code> . Try rebooting and running <code>format</code> again. If the error persists, contact your service representative.
Error: unexpected ioctl error from <code>xy</code> driver.	This error indicates an internal inconsistency in <code>format</code> . Try rebooting and running <code>format</code> again. If the error persists, contact your service representative.
Error: unexpected null partition list.	This error indicates an internal inconsistency in <code>format</code> . Try rebooting and running <code>format</code> again. If the error persists, contact your service representative.
Error: unknown address space type encountered.	This error can occur when you execute the <code>disk</code> command. It means that the definition for the type of address space the controller uses (e.g. VME24D16) is not currently defined within the Sun architecture. This should never happen on a Sun supported controller.
Error: unknown input type.	This error indicates an internal inconsistency in <code>format</code> . Try rebooting and running <code>format</code> again. If the error persists, contact your service representative.
Error: unknown severity value from <code>xd</code> driver.	This error indicates an internal inconsistency in the SunOS kernel. Try rebooting and running <code>format</code> again. If the error persists, contact your service representative.
Error: unknown severity value from <code>xy</code> driver.	This error indicates an internal inconsistency in the SunOS kernel. Try rebooting and running <code>format</code> again. If the error persists, contact your service representative.
Expecting <code>'</code> , found <code><string></code> -- line <i>N</i> of data file.	This error can occur when <code>format</code> scans the data file. It means that there is a syntax error in the data file at the specified line number. For a complete explanation of the data file syntax, see the <i>The format.dat File</i> section.
Expecting <code>:</code> , found <code><string></code> -- line <i>N</i> of data file.	This error can occur when <code>format</code> scans the data file. It means that there is a syntax error in the data file at the specified line number. For a complete explanation of the data file syntax, see the <i>The format.dat File</i> section.
Expecting <code>=</code> , found <code><string></code> -- line <i>N</i> of data file.	This error can occur when <code>format</code> scans the data file. It means that there is a syntax error in the data file at the specified line number. For a complete explanation of the data file syntax, see the section <i>The format.dat File</i> .
Expecting keyword, found <code><string></code> -- line <i>N</i> of data file.	This error can occur when <code>format</code> scans the data file. It means that there is a syntax error in the data file at the specified line number. For a complete explanation of the data file syntax and a list of the keywords, see the <i>The format.dat File</i> section.

- Expecting value, found
<string> -- line *N* of data file. This error can occur when `format` scans the data file. It means that there is a syntax error in the data file at the specified line number. For a complete explanation of the data file syntax, see the *The format.dat File* section.
- Extraction failed. This error can occur when you are executing the `original` or `extract` commands. It means that `format` was unable to extract a complete defect list from the *current disk*. The error messages which precede this one may shed light on the problem. You can create a new defect list by using the `add` command.
- Format failed. This error can occur when you execute the `format` command. It means that `format` could not format the *current disk*. The error messages which precede this one should shed light on the problem.
- Hard sector count less than
nsect. This error can occur when executing either the `format` or `repair` commands on an SMD drive. It means the value specified in the `disk type` for the number of sectors per track is greater than the value claimed by the controller. Check your numbers to make sure they agree with what the controller expects. Also check to be sure the *current disk type* is set correctly.
- Incomplete specification -- line
N of data file. This error can occur during `format`'s processing of the data file. It means that `format` has found one or more missing parameters from the specified line in the data file. For a complete explanation of the data file syntax and a list of required parameters, see the *The format.dat File* section.
- Must specify disk and type as
well as partition. This error can occur when you specify a *partition table* in the command line. It means that you must specify the disk and type as well as the *partition table*, so `format` will know the name and type of the target disk. Reenter the command line and continue.
- Must specify disk as well as
type. This error occurs when you specify the disk type on the command line. It means that you must specify the disk as well as the type. Reenter the command line and continue.
- No current disk. This message is displayed when you execute the `current` command without having first set the *current disk*. You can set the current disk with the `disk` command.
- No default available, cannot
label. This error can occur when you run the `label` command without having specified a *current partition table*. It means that `format` tried to use the default partition table from the data file but could not find one. Define the *current partition table* and rerun the `label` command.
- No default for this entry. This error can occur when you use `format` interactively. It means that you entered a carriage return when there was no default entry for the current input. Specify the value explicitly.
- No defects to delete. This error can occur when you run the `delete` command. It means that the *working defect list* has no entries. Either there are actually no defects in the list,

	or the <i>working defect list</i> is not initialized.
No defined partition tables.	This error can occur when you execute the <code>load</code> command. It means that there are no pre-defined partition tables for the <i>current disk type</i> . You can create a partition table for the disk type by using other commands in the <i>partition menu</i> .
No disks found!	This error can occur during <code>format</code> 's startup processing. It means that <code>format</code> can't find any of the disks in its search path. Make sure you are logged in as the superuser. Also check to be sure the <code>/dev</code> entries for the disks exist, and use <code>MAKEDEV</code> to create them if necessary.
No working list defined.	This error can occur when you execute the <code>print</code> or <code>dump</code> commands with no <i>working defect list</i> defined. You can define a <i>working defect list</i> with the other commands in the <i>defect menu</i> .
Operation on mounted disks must be interactive.	This error occurs when you execute the <code>disk</code> command from a command file and the disk you select has mounted partitions. <code>format</code> will not let you proceed under these circumstances. Note: although this is permitted in interactive mode, it is strongly recommended that you never run <code>format</code> on a disk that has mounted partitions unless you <i>have to</i> .
Repair failed.	This error can occur when executing commands from the <i>analyze menu</i> , as well as from the <code>format</code> and <code>repair</code> commands. It means that the attempted repair of a sector has failed. This message should always follow a more specific error message. Refer to the documentation for the other message(s) to diagnose the problem.
Search path redefined -- line <i>N</i> of data file.	This error can occur when <code>format</code> scans the data file. It means that the search path at line ' <i>N</i> ' is not the only definition in the data file. Remove the extra definition(s) and restart <code>format</code> .
Specified table <i><input></i> is not a known table.	This error can occur when you specify a partition table in the command line that <code>format</code> cannot find. Check to make sure that the table you specified really exists.
Specified type <i><input></i> is not a known type.	This error can occur when you specify a disk type in the command line that <code>format</code> does not support. Check to make sure that you typed the disk type correctly.
Unable to find specified disk <i><input></i> .	This error can occur when <code>format</code> cannot find the <i>current disk</i> you have specified. Check to make sure that you have typed the disk name correctly. You can use the <code>disk</code> command to get a list of the disks <code>format</code> has found.
Unable to find suitable alternate sector.	This error can occur when you are repairing a defective sector on an SMD disk. This can happen from <code>repair</code> , <code>format</code> or any of the surface analysis commands. It means that <code>format</code> was not able to find an alternate sector to replace the defective one. This is usually because mapping is not allowed on certain areas of the disk. There is nothing you can do about this.

- Unable to get tty parameters. This error can occur when `format` fails in an attempt to obtain the current parameters from the terminal you are running from. This generally means that something is seriously wrong. Reset your terminal and try rerunning `format`.
- Unable to locate defect #N. This error can occur when executing the `format` command on an SMD disk. It means that either `format` could not read the headers for the track containing the defect, or that the header information returned was garbage. In general, this means that a hardware error has occurred on the controller and/or the disk. Run hardware diagnostics to find the problem.
- Unable to locate defect. This error can occur when you are repairing a defective sector on an SMD disk. It means that either `format` could not read the headers for the track containing the defect, or that the header information returned was garbage. In general, this means that a hardware error has occurred on the controller and/or the disk. Run hardware diagnostics to find the problem.
- Unable to open '/dev/tty'. This error can occur when `format` tries to open the terminal special device file. It indicates a problem in the SunOS kernel. Try rebooting and running `format` again. If the error persists, contact your service representative.
- Unable to open command file <input>. This error can occur when `format` attempts to open the command file you have specified. Check to make sure that the file exists, that you have permission to read it, and that the filename has been specified correctly.
- Unable to open data file <input>. This error can occur when `format` attempts to open the data file you have specified. Check to make sure that the file exists, that you have permission to read it, and that the filename has been specified correctly.
- Unable to open defect file <input>. This error can occur when executing the `dump` or `load` commands. It means that `format` failed to open the defect file you specified. Check to make sure that the filename has been specified correctly.
- Unable to open log file <input>. This error can occur when `format` attempts to open the log file you have specified. Check to make sure that the filename has been specified correctly.
- Unable to open mount table. This error can occur when `format` checks to see if the *current disk* has any mounted partitions on it. This error shows an internal inconsistency in the SunOS kernel. Try rebooting and running `format` again. If the error persists, contact your service representative.
- Unable to read drive configuration. This error can occur when executing the `format` or `repair` commands on an SMD disk. It means that the controller cannot read the configuration information from the disk. In general, this means that a hardware error has occurred on the controller and/or the disk. Use diagnostics to find the problem.
- Unable to read track headers. This error can occur when executing the `format` or `repair` commands on an SMD disk. It means that the controller cannot read the track headers from the disk. In general, this indicates that a hardware error has occurred on the

	controller and/or the disk. Use diagnostics to find the problem.
Unable to repair defect # <i>N</i> .	This error can occur when executing the <code>repair</code> command. In general, this indicates that a hardware error has occurred on the controller and/or the disk. Check the cabling and, if necessary, run hardware diagnostics to find the problem.
Unable to repair track headers.	This error can occur when executing the <code>repair</code> command on an SMD disk. It means that <code>format</code> could not write back the repaired header for the track containing the defect. In general, this indicates that a hardware error has occurred on the controller and/or the disk. Use diagnostics to find the problem.
Unable to set tty parameters.	This error occurs when <code>format</code> fails in an attempt to reset your terminal parameters. This error indicates an internal inconsistency in the SunOS kernel. Try rebooting and running <code>format</code> again. If the error persists, contact your service representative.
Unable to write track headers.	This error can occur when executing the <code>format</code> or <code>repair</code> commands on an SMD disk. It means that <code>format</code> could not write a track header to the <i>current disk</i> . In general, this indicates that a hardware error has occurred on the controller and/or the disk. Use diagnostics to find the problem.
Value <i><input></i> is not a known ctlr name -- line <i>N</i> of data file.	This error occurs when the data file contains a reference to an unknown controller. Check to make sure that the spelling is correct and that the controller is supported by the <code>format</code> program. For a complete explanation of the data file syntax and a list of supported controllers, see the <i>The format.dat File</i> section.
Value <i><input></i> is not a known disk name -- line <i>N</i> of data file.	This error occurs when the data file contains a reference to an unknown disk type. Check to make sure that the spelling is correct and that the disk type is supported by the <code>format</code> program. For a complete explanation of the data file system, see the <i>The format.dat File</i> section.
Value <i><input></i> is not an integer -- line <i>N</i> of data file.	This error occurs when <code>format</code> is parsing the data file. It means that there is a syntax error in the data file at the specified line number. For a complete explanation of the data file syntax, see the <i>The format.dat File</i> section.
Warning: disk not set for slip-sectoring.	This warning can occur when you execute the <code>repair</code> , <code>extract</code> , or <code>format</code> commands on an SMD disk. It means that the <i>current disk</i> is not set up for slip-sectoring. It is strongly recommended that you set up all the disks for slip-sectoring. See the disk and controller manuals for more information.
Warning: error saving bad block map table.	This warning can occur when executing the <code>format</code> and <code>repair</code> commands on an SMD disk. It means that <code>format</code> attempted to write the bad block map table to the <i>current disk</i> but failed. In general, this indicates a hardware error. Use diagnostics to find the problem.

Warning: error saving defect list.

This warning can occur when executing the `format` and `repair` commands. It means that `format` attempted to write out the defect list to the *current disk* but failed. In general, this indicates a hardware error. Use diagnostics to find the problem.

Warning: error telling SunOS bad block map table.

This error occurs when `format` tries to update the bad sector mapping information for an SMD disk. This error indicates an internal inconsistency in the SunOS kernel. Try rebooting and running `format` again. If the error persists, contact your service representative.

Warning: error telling SunOS drive geometry.

This error occurs when `format` tries to update the drive geometry information for the *current disk*. This error indicates an internal inconsistency in the SunOS kernel. Try rebooting and running `format` again. If the error persists, contact your service representative.

Warning: error telling the SunOS drive type.

This error occurs when `format` tries to update the drive type information for an SMD disk. This error only occurs on the Xylogics 450/451 controller. This error indicates an internal inconsistency in the SunOS kernel. Try rebooting and running `format` again. If the error persists, contact your service representative.

Warning: error telling SunOS partition table.

This error occurs when `format` tries to update the partition table information for the *current disk*. This error indicates an internal inconsistency in the SunOS kernel. Try rebooting and running `format` again. If the error persists, contact your service representative.

Warning: error writing backup label.

This warning can occur when executing the `format` or `label` commands, as well as several of the surface analysis commands. It means that an attempt to write the backup disk labels for the *current disk* failed. In general, this indicates a hardware error. Use diagnostics to find the problem.

Warning: error writing primary label.

This warning can occur when executing the `format` or `label` commands, as well as several of the surface analysis commands. It means that an attempt to write the primary disk label for the *current disk* failed.

Warning: unable to pinpoint defective block.

This warning is issued from several commands in the *analyze menu*, as well as from the `format` command. It means that surface analysis encountered a media related error, but was unable to pinpoint exactly which sector is defective because the error did not reoccur upon closer inspection. You can repair the sector in the error message if you want to be safe, or you can wait until the defect becomes more evident.

Warning: unable to restore original data.

This warning is issued from the `test` command. It means that the original data on the disk was overwritten during a scan of the disk and that the attempt to rewrite the original data failed. This message should be accompanied by other messages which will illuminate the cause of the failure.

Warning: unable to save defective data.	This warning can occur when executing <code>format</code> , <code>repair</code> , or any of the surface analysis commands. It means that <code>format</code> repaired a bad sector and tried to save the data in it, but failed. There is not much that can be done under these circumstances except to restore the damaged file from backup media.
Warning: unable to save track data.	This warning can occur when executing the <code>format</code> , <code>repair</code> , or any of the surface analysis commands. It means that <code>format</code> repaired a bad sector and found that it was unable to save the data for the rest of the track past the bad sector. This may mean that the rest of the track is bad as well. At this point, you should backup the disk and do surface analysis to test the disk media.
Warning: working list modified but not committed.	This warning can occur when you exit the <i>defect menu</i> . It means that you have modified the <i>working defect list</i> but have not committed the list. To set the <i>current defect list</i> equal to the <i>working defect list</i> , run the <code>commit</code> command.
Working list was not modified.	This warning can occur when you execute the <code>restore</code> command to set the <i>working defect list</i> equal to the <i>current defect list</i> . It means that no changes have been made to the <i>working defect list</i> since the last <code>restore</code> command, so the <i>working defect list</i> is already up to date.
<i>N</i> is out of range.	This error can occur whenever <code>format</code> is expecting input. It means that a value given to the program was out of the bounds specified for the given parameter. To see the bounds of any input, enter a question mark. Then simply enter the correct value.
<i>N/N/N</i> is out of range.	This error can occur whenever <code>format</code> is expecting input. It means that a value given to the program was out of the bounds specified for the given parameter. To see the bounds of any input, enter a question mark. Then simply enter the correct value.
<i><input></i> is ambiguous.	This error can occur any time <code>format</code> is expecting input. It means that the input specified is not sufficient to identify the choice made. To see the choices for any input, simply enter a question mark.
<i><input></i> is not an integer.	This error can occur any time <code>format</code> is expecting input. It means an integer is required, but something else was specified. Correctly reenter the input.
<i><input></i> is not expected.	This error can occur any time <code>format</code> is expecting input. It means the input specified does not match any of the choices. To see the choices available, simply enter a question mark.
Errors for SCSI Drives	These errors pertain specifically to SCSI drives only.
mode select failed.	This error occurs when running the <code>format</code> command on a SCSI ACB-4000 controller. The <code>MODE SELECT</code> command, which is issued prior to the <code>FORMAT</code> command, has failed.

SCSI MD21 controller does not support the 'show' command.	The SCSI MD21 controller does not have the necessary software support to use the show command. Thus, 'show' is inoperable while using that controller.
unexpected ioctl error from sd	This error indicates an internal inconsistency in format. If the error persists, contact your service representative.
Block N (), [Fatal Recovered Corrected] [media error non-media error] (<i><message></i>) [during <i><where></i>]	This message contains some additional information on a SCSI command error.
Could not translate logical defect to bfi	This message occurs when running the add and load commands from the defect menu, the repair and format commands, and many of the commands in the analyze menu. The ACB-4000 controller failed to correctly translate a logical sector to a bfi (Bytes From Index) format.
retries= N (M)	This message occurs when extended messages are enabled while formatting a SCSI disk. It shows the requested and default disk error retry count.
Warning: Using default error recovery params.	This message occurs when formatting a SCSI disk and extended messages are enabled. It indicates that the disk's default error recovery parameters will be used, and not the Page 0x1 MODE SELECT parameters previously requested.
inactivity limit= N (M)	This message occurs when formatting a SCSI disk and extended messages are enabled. It shows the disconnect/reconnect requested and default parameters.
Warning: Using default disconnect/reconnect parameters.	This message occurs when formatting a SCSI disk and extended messages are enabled. It indicates that the disk's default disconnect/reconnect parameters will be used and not the Page 0x2 MODE SELECT parameters previously requested.
Warning: Using default drive geometry.	This message occurs when formatting a SCSI disk and extended messages are enabled. It indicates that the disk's default geometry will be used, and not the Page 0x4 MODE SELECT parameters previously requested.
cylinders= N (M) heads= H (J)	This message occurs when formatting a SCSI disk and extended messages are enabled. It shows the requested and default cylinders and heads.
cache mode= $0xX$ ($0xY$) min. prefetch multiplier= P max. prefetch multiplier= M threshold= T (Z) in. prefetch= N (I) max. prefetch= Q (R)	This message occurs when formatting a SCSI disk and extended messages are enabled. It shows the requested and default parameters for cache control.
Warning: Using default cache params.	This message occurs when formatting a SCSI disk and extended messages are enabled. It indicates that the disk's default cache parameters will be used, and not the Page 0x38 MODE SELECT parameters previously requested.

Warning: Using default mode
select parameters.

Warning: Drive format may not
be correct.

This message occurs when formatting a SCSI disk. It indicates that the disk format parameters as specified through the format.dat file or type command were not accepted by the controller due to a Page 0x3 MODE SELECT failure, so the format of the disk drive may be incorrect.

can't zalloc defect list memory
can't malloc defect list memory

These messages occur when formatting a SCSI disk. This error happens when there is not enough memory to satisfy a request by format for temporary storage. If you are running format online, try to re-run it when there is less activity on the system.

Warning: Using internal drive
defect list only.

This message occurs when formatting a SCSI disk. It indicates that the drive rejected the supplied defect list. The format command will be retried using only the drive's internal (manufacturer's) defect list.

No grown defect list.

This message occurs when extended messages are enabled. It indicates a failure reading the grown defect list from the disk, using the original command. An attempt to read the manufacturer's defect list will be attempted next.

No manufacturer's defect list.

This message occurs when running the original command, and indicates a failure reading the manufacturer's defect list from the disk. A null defect list will be returned.

Adding Hardware to Your System

This chapter explains many aspects of adding hardware to your system. It covers how to add a new board to the system, how to add peripheral devices like terminals and modems to a serial port, and how to add a printer to the system.

In most cases adding new hardware to your system is a three-step process

1. Connect the actual hardware piece as appropriate.
2. Reconfigure the kernel, if necessary.
3. Edit the necessary files on your system to adapt it to the new device. For each kind of new hardware device, the process is explained in the appropriate section below.

The first kind of hardware device covered is a circuit board. You will probably have to reconfigure your kernel when a board is added.

Next, a section explains general procedures for adding devices to asynchronous serial ports, followed by specific instructions for terminals and modems.

Finally, the chapter explains how to add a printer. You can hook up printers on a serial port or connect them directly to a controller board that drives a given model; each case is covered.

11.1. Adding a Board to Your System

When you add a new board to any Sun system, you need to do at least three tasks

1. Insert the board itself in the computer's card cage.
2. Add a device driver (a group of procedures or routines) to the kernel.
3. Modify the file system to accept the new device.

This section explains the last two procedures: adding a device driver and modifying the file system. For each board Sun supports, there is an explanation of card cage installation in the *Hardware Installation Manual* for your Sun system.

Kernel Modification

Reconfiguring the kernel was fully described in the chapter *Reconfiguring the System Kernel*. This next section explains how to modify an existing kernel in order to add new equipment.

Kernel Configuration for Servers

The kernel device drivers help the kernel pass information between the system hardware and the system software. When you add a new board to your computer, you also need to add or enable the line in the kernel configuration file that describes the particular device driver that provides the interface between the kernel and the board. Then, when you reconfigure the new kernel, the device driver program is included in the kernel. Remember that the more device drivers you include in the kernel, the more space in main memory the kernel will take up. Therefore, make sure the kernel configuration file enables only the device drivers needed by the equipment on your system.

Below is an example of an entry in the kernel configuration file for a Sun-3 color board. To add a color board device driver to the kernel, include this line in the kernel configuration file.

```
device cgtwo0 at vme24d16 ? csr 0x400000 priority 4
vector cgtwointr 0xa8
```

The chapter *Reconfiguring the System Kernel* contains a copy of GENERIC kernel configuration files for Sun-3, Sun-3x, Sun-4 and Sun SPARCstation machines. For further explanation, each device listed in `/usr/share/sys/sun[3,3x,4,4c]/conf/GENERIC` is documented in Section 4, Special Files, in the *SunOS Reference Manual*. For example, if you look up `cgtwo` in that manual, you would find out, among other things, what values are mandatory for specified device driver fields.

For boards that Sun does not support, you will need to write your own device driver and place an entry for it in the configuration file. This procedure is described in *Writing Device Drivers*. It requires expert understanding of the kernel. You can often obtain device drivers from the board manufacturer or distributor.

When you have determined the device driver information for the new board to add to the kernel configuration file, you need to do the steps shown below to complete the kernel reconfiguration process. These steps explain the process on a system named "grendel"

Note: The following procedure should be used only if the device you are adding already has a driver installed in the kernel. If you are adding new hardware that goes with a new driver, you must also edit the `conf.c` file and the `/usr/share/sys/sun[3,3x,4,4c]/conf` file, as explained in the *Writing Device Drivers* manual.

1. Go to the `/usr/share/sys/sun[3,3x,4,4c]/conf` directory. Make a copy of the current kernel configuration file to keep until the new one is installed and running.

```
# cd /usr/share/sys/sun[3,3x,4,4c]/conf
# cp GRENDEL old.GRENDEL
```

If you have not built your own kernel, you must base your kernel configuration on the GENERIC kernel configuration file, and prepare for building your new kernel

```
# cd /usr/share/sys/sun[3,3x,4,4c]/conf
# cp GENERIC GRENDEL
# chmod +w GRENDEL
```

2. Edit the kernel configuration file, adding a device driver entry for the board you have installed into the card cage. Look in Section 4 of the *SunOS Reference Manual* for specifications for the boards that Sun supports.
3. Run `/usr/etc/config` on the new kernel configuration file.

```
# /usr/etc/config GRENDEL
```

4. Build the new kernel.

```
# cd ../GRENDEL
# make
```

5. Now install the new kernel and boot your machine by doing the following
First move the original working kernel to another (safe) place. Then copy the new kernel to the place of the original. Finally boot up the system with this new kernel.

```
# mv /vmunix /vmunix.old
# mv vmunix /vmunix
# /etc/halt
```

The system goes through the halt sequence, then the monitor displays its prompt, at which point you can boot the system by typing **b** at the monitor prompt.

The system boots up multiuser, and then you can try things out.

6. If the new kernel does not seem to function properly, halt the system and boot from the original kernel. Then reinstall the original kernel. Once you are booted up on the original, you can go about trying to fix the faulty kernel.

Adding Devices

You must be superuser to make the following system modifications.

The kernel communicates with devices through a special file in the directory `/dev`. When a new board is added to the system, you probably make a new entry for it in the `/dev` directory. This is relatively easy to do with a shell script called `MAKEDEV`, located in `/dev`. `MAKEDEV` takes an argument that is the device-name of a device supported by the system, and automatically installs the files for that device in the `/dev` directory. Note, however, that some boards do not require an entry in the `/dev` directory, such as ethernet boards. Check the following tables to see if a `MAKEDEV` argument is listed for the appropriate board.

If you are adding a Sun-4 color board, for example, you need to run MAKEDEV like this:

```
# cd /dev
# MAKEDEV cgnine0
```

The following table gives a list of the arguments MAKEDEV accepts for the boards supported by Sun. It also includes the name of the device driver that is listed into the kernel configuration file — the MAKEDEV argument and the device driver name are often identical, but in several cases are different so be attentive. Where an asterisk appears, it means that a logical number should replace it — '0' being the first, '1' being the second, and so forth.

Table 11-1 Sun Supported Tape Controller Boards

MAKEDEV Argument	GENERIC File Device Driver	Description of Device
st*	st*	Sysgen SC-4000 1/4" SCSI to QIC 02 Controller
st*	st*	Emulex MT02 1/4" SCSI to QIC 36 Controller
N/A	tm*	1/2" 1600bpi, CPC Tapemaster
mt*	mt*	40MB 9 track reel, CDC 92181 (Keystone)
tm*	tm*	1/2" 1600bpi, CPC Tapemaster 40MB 9 track reel, CDC 92181 (Keystone)
N/A	xtc*	1/2" Xylogics 472
xt*	xt*	150MB 9 track reel, 6250bpi Fujitsu M244X
N/A	xtc*	1/2" Xylogics 472 40MB 9 track reel, 1600 bpi CDC 92181
		150MB 9 track reel, 6250bpi Fujitsu M244X

Table 11-2 Sun Supported Disk Controllers

MAKEDEV Argument	GENERIC File Device Driver	Description of Device
N/A	xy*	SMD Controller, Xylogics 450/451
xy*	xy*	SMD Disk connected to a Xylogics 450/451
N/A	xdc*	SMD Controller, Xylogics 7053
xd*	xd*	SMD Disk connected to a Xylogics 7053
N/A	sc*	SCSI-2 Host Adapter
N/A	si*	SCSI-3 Host Adapter
N/A	sw*	SCSI Host Adapter on the 4/110
N/A	sm*	SCSI-ESP Host Adapter on the 4/330
sd*	sd*	Adaptec SCSI Disk

Table 11-3 *Sun Supported Terminal Multiplexor Devices*

MAKEDEV Argument	GENERIC File Device Driver	Description of Device
mti*	mti*	System MTI-800/1600 or Sun ALM
mcp*	mcp*	Sun ALM-2 or Sun Multiprotocol COMM processor

Table 11-4 *Sun Supported Ethernet Controller Devices*

MAKEDEV Argument	GENERIC File Device Driver	Description of Device
N/A	ie*	Intel Ethernet Board
N/A	le0	LANCE Ethernet board

Table 11-5 *Sun Supported Printer Devices*

MAKEDEV Argument	GENERIC File Device Driver	Description of Device
pp*	pp0	Paradise parallel port
vpc*	vpc*	Versatec & Centronics (System VPC-2200)

Table 11-6 *Sun Supported Graphics/Windows Devices*

MAKEDEV Argument	GENERIC File Device Driver	Description of Device
cgtwo*	cgtwo*	8-bit color frame buffer
cgfour*	cgfour*	8-bit color frame buffer w/overlay
cgsix*	cgsix*	8-bit color accelerated buffer
cgeight*	cgeight*	24-bit color frame buffer
cgnine*	cgnine*	24-bit color accelerated frame buffer
bwtwo*	bwtwo*	Sun-3 monochrome graphics device
gpone*	gpone*	Graphics processor board

Table 11-7 *Miscellaneous Sun Supported Devices*

MAKEDEV Argument	GENERIC File Device Driver	Description of Device
N/A	sc*	SCSI-2 Host Adapter
N/A	si*	SCSI-3 Host Adapter
N/A	sw*	SCSI Host Adapter on the 4/110
N/A	sm*	SCSI-ESP Host Adapter on the 4/330
fpa	fpa	Sun Floating Point Accelerator board
des	des	DES Encryption chip
taac	taac	Applications Accelerator

Trouble Shooting

Here is a general list of hints for debugging new boards.

1. Check the hardware. Re-seat the board to assure electrical contact. Check hardware manuals to make sure all switch and jumper settings are proper.
2. If you boot the system with a copy of the GENERIC kernel after installing a device that Sun supports, and you still have the problem, it is probably a hardware problem. If the problem disappears using GENERIC, look in the kernel or in /dev for the problem.
3. Make sure the new device is included correctly in the kernel configuration file. One way to ascertain this is to look in a file called /var/adm/messages, which collects all the messages generated during boot-up. By looking at the last boot-up listing in /var/adm/messages, you can determine positively whether the device was included in the kernel and was one of one of the devices the boot-up procedure found.
4. Do an `ls -l` in the /dev directory to make sure the new device is installed. The `ls -l` will also show the permissions and major and minor numbers for each device present. Check the permissions for the device against those found in the section of the MAKEDEV script that installs the device in question; they should match. Also make sure the device major and minor numbers match those in MAKEDEV.
5. Try the simplest commands to see if ANYTHING is working. The closer you come to isolating the spot where the system breaks down, the fewer places you need to look to make fixes. For example, see if a new printer board works by simply sending a file to it using `cat`:

```
% cat myfile > /dev/lp
```

This is better than trying to pipe a job through text processors to `lpr`, where the job could fail for a reason that has nothing to do with the new board.

11.2. Connecting Devices to Asynchronous Serial Ports

Please read the section on “Asynchronous Serial Ports” in the *Hardware Installation Manual* for your machine before proceeding further. There you will find a discussion of serial port signals for your machine. Become familiar with the hardware specifications for your Sun model.

General Theory

This section introduces the general topic of connecting equipment to serial ports and discusses some of the terminology. For specific instructions on adding terminals, modems or printers, see those sections below.

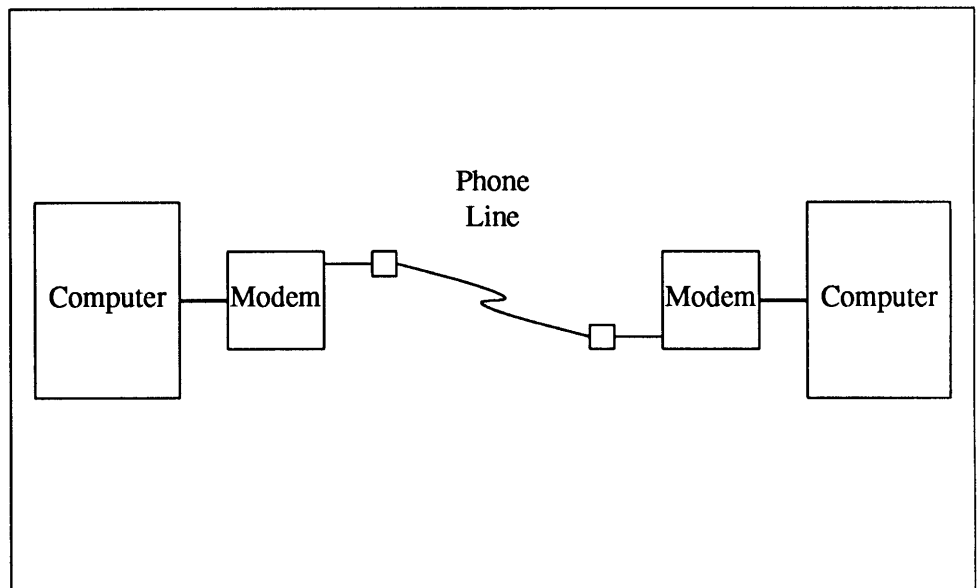
Put simply, a serial port sends a byte of information bit by bit over a single line. All of the Sun serial ports have RS-232-C cabling conventions, but use RS-423 communications signal conventions. You can attach modems, terminals, printers, plotters, or other peripheral serial devices that accept the RS-423 signaling to the serial port connectors on your machine. All ports on Sun machines are wired as DTE — data terminal equipment. This means that the data transmit signal from the port is on pin 2 and the receive data from the peripheral is on pin 3. To connect a Sun workstation directly to a terminal, printer, computer, or other DTE device, use a standard “null modem” cable that crosses lines 2 and 3,

thereby enabling the proper signal connection at the other end.

Some serial interfaces require special additional wiring. For example, the MTI board requires pin 5 to be asserted. This can be satisfied if the peripheral supplies an appropriate signal; otherwise, pin 4 and 5 can be connected at the MTI connector end of the cable. Check the documentation that came with the peripheral to see if any special wiring is needed.

The Sun workstation is connected to phone lines by adding a modem. You can connect the modem directly to the serial port. Modems are wired as DCE (data communication equipment) devices. A DCE receives serial data from the DTE on pin 2 and sends it down the telephone line. A DCE receives telephone data and sends it out on pin 3 to the DTE. See 11-1 below for a depiction of how two computers would transmit signals through modems along cables and phone lines.

Figure 11-1 *Communications Through Modems*



When you connect a peripheral device to a serial port on the workstation, make sure the cable connection between the serial port and the device is properly installed. Next make sure the appropriate serial device is configured in the kernel. Note that the GENERIC kernel contains all the Sun supported device drivers. If you have reconfigured or customized the kernel, you may need to add back the appropriate serial driver. See chapters *Reconfiguring the System Kernel* and *Advanced Administration Topics* for more information about kernel configuration. If specific kernel changes are necessary, they are discussed in the following sections.

General Debugging Hints

In this section is a discussion of two general areas where problems occur when you have added new peripheral devices to the system.

Hardware Changes — If you have added new hardware and there are problems, check the cabling and connections first. Is everything tight? Once again verify that the new device accepts the RS-423 communication protocol, and consult the product manufacturer's manual to make sure it has been installed properly. Are the workstation and the new device set up for compatible communication? The Sun defaults to the following: 7 data bits, even parity, 9600 baud rate, flow control enabled (XON/XOFF). Check which control signals the other equipment expects.

Next, check the condition of the cable. Here you may want to refer once again to your Sun Workstation *Hardware Manual* to determine what signals are sent from and received at specific ports on your board. To check for cable problems it is helpful to have a *breakout box*. A breakout box plugs into the RS-232 cable, providing a patch panel that allows you to connect any pin to any other pin(s); it will often contain LEDs, which display the presence or absence of a signal on each pin. These tools can be purchased through any computer supplier. Sometimes a cable has become corrupt and signals are being flipped randomly. In this case a line may receive when it should transmit, or the reverse. These suggestions should give you a start on troubleshooting. Problems specific to terminals are covered in the *Adding Terminals* section below.

New Software — If you have problems after installing a new system kernel along with the new peripheral device, there could be something wrong with the new kernel. This is likely to be a total non-response symptom rather than the intermittent glitches of a bad line. Rename the new kernel to something other than `/vmunix` and install the `GENERIC` kernel in its place, then boot the system with `GENERIC`. If the problem disappears with `GENERIC`, you have a problem with the new kernel — possibly you did not add the device driver, or did not add it properly. If the problem persists with `GENERIC`, it is probably in the cabling or the hardware.

11.3. Adding a Terminal to Your System

Please read the previous section called *Connecting Devices To Asynchronous Serial Ports* before proceeding further. The general discussion there will prepare you to install a new terminal on your system.

This section explains the steps necessary for adding a terminal, and gives some hints on what to do if you run into problems.

Serial Port and Cable Connection

Make sure you have an available serial port on your workstation. The number of serial ports varies from one Sun model to another. Each Sun workstation provides at least two asynchronous serial ports controlled by the Serial Communications Controller on the Sun CPU board (with the exception of the 386i, which has only 1.) If all the existing serial ports are in use, and you are not planning to disconnect any current peripheral devices to make a port available, you will need to add a new board to the card cage to increase the number of serial ports. The instructions for adding the board were given earlier in the appropriate board installation guide.

When you do have a port available, connect the terminal to the workstation with a null modem cable. A basic null modem cable swaps lines 2 and 3 so that the proper transmit and receive signals are communicated between two Data Terminal Equipment (DTE) devices. Line 7 goes straight through.† Make sure all the connections are secure and check control signals.

Kernel Modification

You must be superuser to make kernel modifications. If you are adding a first terminal, make sure that the system kernel contains a `zs0` device driver to run the CPU ports (this is standard). If you are adding terminals to a Sun ALM or a Systech multi-terminal interface board (MTI), make sure the proper `mti` driver entry is in the kernel configuration file. If you are adding devices to an ALM-2 board, make sure the proper `mcp` driver entry exists in the configuration file.

If you had to add a new board to make ports available for your terminals, you probably made necessary kernel changes already, as outlined previously in the section *Adding A Board To Your System*.

Look at the system diagnostic messages in `/var/adm/messages` to make sure the system is configured the way you want it.

Look in your kernel configuration file in the `/usr/share/sys/sun[3,3x,4,4c]/conf` directory to find the device drivers for which your system is configured. An explanation of the device driver entries in this file is given in chapters *Reconfiguring the System Kernel* and *Advanced Administration Topics*. For terminal connection, make sure the `flags` bit is set for a hard-wired line on the port — set to `on`, or that you have set up the support files to take advantage of the `ttysoftcar` command.

Adding Devices

The SunOS operating system needs a special file in `/dev` to communicate with any device. For serial ports, these are known as `/dev/tty*` — where the asterisk is a letter or number value or both. When you add a board, you have to run the `MAKEDEV` shell script to install the special files for it in `/dev`. If you are connecting to available ports on a particular board already in use, then `MAKEDEV` has already been run, and you can ignore the following information.

The `MAKEDEV` command takes an argument that is the device name for the device in the kernel. In the case of a serial port board, one `/dev/tty*` file is created for each terminal interface on the board. The table below shows examples of the `MAKEDEV` command for the boards that Sun supports, and the names of the entries `MAKEDEV` creates in `/dev`.

† The Systech MTI board requires pin 5 asserted. The simplest way to get this is to connect pins 4 and 5 at the connector going to the MTI port.

Table 11-8 Using MAKEDEV For New Boards

Type	Command	Devices
Sun CPU Board	MAKEDEV std	ttya,ttyb
Sun 4/330 Board	MAKEDEV std	ttya — ttyd
Systech MTI-800 Board	MAKEDEV mti0	tty00 — tty07
Systech MTI-1600 Board	MAKEDEV mti0	tty00 — tty0f
Sun ALM-2 Board	MAKEDEV mcp0	ttyh0 — ttyhf

Next you will need to edit `/etc/ttytab`, the file that tells the system about the new terminal. This file is read by the `init` process and specifies which serial ports will have a login process created for them.

There should be one line in `/etc/ttytab` for every serial port.* The name of the port in `/etc/ttytab` is the same as its name in `/dev`. The `/etc/ttytab` file looks in part like this:

```
#
# name  getty                type          status  comments
#
console "/usr/etc/getty std.9600"  sun          on local secure
ttya    "/usr/etc/getty std.9600"  925         on local
ttyb    "/usr/etc/getty std.9600"  unknown     off local
.
.
.
ttyd0   "/usr/etc/getty Fast-Dial-1200" dialup       on remote
```

A comment begins with the character “#” and continues to the end of the line.

The first field of a line is the name of the device in `/dev`.

The second field is the program that `init` should run on this line, together with all its arguments. Normally, `/usr/etc/getty`, which performs such tasks as setting the baud rate, reading the login name, and calling `login`, is run. Its argument is the name of the baud rate code from `/etc/gettytab` to be used. Looking in the file `/etc/gettytab` you will find the values your system recognizes for different baud rates. For example:

```
c|std.300|300-baud:\
      :nd#1:cd#1:sp#300:
f|std.1200|1200-baud:\
      :fd#1:sp#1200:
2|std.9600|9600-baud:\
      :sp#9600:
```

There are also `ttyp*`, `ttyq*`, `ttyr*`, and `ttys*` lines. These are required for the window system and other programs and must appear as installed by the installation procedure, except that the “secure” status may be changed.

are the `/etc/gettytab` entries for initialization of terminals at three different baud rates. Therefore, “`std.9600`” as the argument of `getty` would set the rate to 9600, and “`std.1200`” would set it to 1200.

The third field is the type of terminal on that port. To find the kind of terminal types your system recognizes, look in the file `/usr/share/lib/termcap` (see `termcap(5)`). This is a database describing the capabilities of terminals and the way operations are performed on them. If your terminal type is found in `/usr/share/lib/termcap`, use it. If your new terminal has an emulation mode for one of the terminal types found in your `termcap` file, put that name into `/etc/ttytab` and set the corresponding emulation mode on the terminal. If your terminal type is not in `/usr/share/lib/termcap`, you may create your own `termcap` entry, as explained in the chapter *Modifying the termcap File*. The type “`dialup`” should be used for phone-in lines. The information in `/etc/ttytab` is read by `login` to initialize the `TERM` environment variable.

The fourth field is either “`on`” or “`off`,” optionally followed by some flag values. If it is “`off`,” `init` ignores that line, and no logins will be able to occur on that line; if it is “`on`,” `init` creates a login process for that line. The flag fields are separated from “`on`” or “`off`” and from each other by spaces. The flag “`secure`” indicates that the terminal is “secure;” You can only log in as the superuser (“`root`”) on “`secure`” terminals. (Note that if the console terminal is not flagged as “`secure`,” when entering single-user mode the system will require the you to type in the superuser password before the shell is started.) The flag “`window=program`” indicates that the specified *program* together with its arguments is to be run by `init` before it runs the program (such as `getty`) that it would normally run. The *program* and its arguments are specified by a quoted string. This is used, for example, if the program that `init` would normally run requires a window system to be started.

The sample `/etc/ttytab` file above indicates that the console is a Sun Workstation console, and is a “`secure`” terminal. The terminal attached to `/dev/ttya` is a Televideo TVI-925, running at 9600 baud, and is not “`secure`”. The terminal type of the terminal attached to `/dev/ttyb` is unknown, and logins are disabled on that terminal. If you were putting a Televideo TVI-925 terminal on `/dev/tty2` at baud rate 9600, you would add the following line to the example `/etc/ttytab` file shown above:

```
ttys2      "/usr/etc/getty std.9600"  925      on
```

at the end of the file. This terminal will have a login process started for it at 9600 baud. After editing the `/etc/ttytab` file you **must** notify `init`, which then reads the new information in `/etc/ttytab`. The command to do this is:

```
# kill -1 1
```

Finally, you need to set some switches on the terminal so it can communicate with the machine. Check to see if you have changed any settings on the Sun from the defaults given here: 7 data bits, even parity, flow control enabled (`xon/xoff`); set baud rate to the same rate you set in the `/etc/ttytab` file.

If possible, set up the terminal to ignore the parity bit.

Problems

If you have problems after installation, check the cabling first, as outlined previously in the section *Connecting Devices to Asynchronous Serial Ports*. Next, check the information in the `/etc/ttytab` file, and make sure you have restarted `init` as explained.

If the terminal behaves strangely, check the `/usr/share/lib/termcap` file. You may have unexpected white space, or you may have forgotten to escape the newline character in your entry: a backslash (`\`) followed immediately by a newline (carriage return) will continue a line onto the next.

If you have not correctly set the tty type in the `/etc/ttytab` file, you can set it at your terminal with the command:

```
# set term=sometype
```

There are several ways to check aspects of your terminal environment using `stty`:

- The `stty` command typed with no argument will tell you the baud rate of the terminal and the settings of options that are different from their defaults.
- `stty` with the `all` argument reports all normally used option settings.
- The command `stty everything` reports everything that `stty` knows about.

The `tset` command with the `-r` argument shows the current terminal type. Additionally, the `printenv` command with no arguments prints out the variables in the environment.

You can type `set` with no arguments, to report the value of all shell variables.

11.4. Adding a Modem to Your System

Please read the section above called *Connecting Devices to Asynchronous Serial Ports* before proceeding further. The general discussion there will prepare you to install a modem on your system.

This section explains the steps necessary for adding a modem, and gives some hints on what to do if you run into problems.

Sun has found that the several modems work well with Sun systems: the Hayes Smartmodem 1200, the USRobotics 2400, and the Telebit TrailBlazer,

Cable Connection and Modem Switches

Cable Connection — Connect the modem to an open serial port on the workstation with an RS-232 cable that has pins 2 through 8 and pin 20 wired straight through. You may also use full 25 pin cable to connect the modem to the system. Make sure all the connections are secure.

Modem Switches — The switch settings shown below work for use with `tip` and `uucp`. Always read the manufacturer's manual before attempting to adjust equipment.

The Hayes Smartmodem 1200

The proper switch settings are given below. There is only one switch panel on the Hayes; down is `on` and up is `off`.

- Switch 1 up for hardware DTR.
- Switch 2 down for numeric result codes.
- Switch 3 down to send result codes.
- Switch 4 down to not echo commands.
- Switch 5 up to answer incoming calls.
- Switch 6 up for hardware Carrier Detect.
- Switch 7 up for connection to RJ11 modular jack.
- Switch 8 down enable command recognition.

After changing the switch settings on any model, you must turn off power to the modem, wait a few seconds, and then power it back up. Looking at the front of a properly installed modem, you should see TR light up when the modem is not in use. If auto answer is enabled (switch #5, above), AA should also be lit. The lights should be lit or blinking when the modem is in use.

The USRobotics Courier 2400

The proper switch settings are given below. There is two switch panels on the USRobotics; a ten switch panel and a single “Quad” switch panel. On the USRobotics modem, down is `on` and up is `off`.

- Switch 1 up for hardware DTR.
- Switch 2 down for numeric result codes.
- Switch 3 down to send result codes.
- Switch 4 down to not echo commands.
- Switch 5 up to answer incoming calls.
- Switch 6 up for hardware Carrier Detect.
- Switch 7 up for connection to RJ11 modular jack.
- Switch 8 down enable command recognition.
- Switch 9 down do not disconnect with +++.
- Switch 10 up not used
- “Quad” Switch up pins 2 and 3 straight through.

After changing the switch settings on the modem , you must turn off power to the modem, wait a few seconds, and then power it back up. Looking at the front of a properly installed modem, you should see TR light up when the modem is not in use. If auto answer is enabled (switch #5, above), AA should also be lit. The lights should be lit or blinking when the modem is in use.

Telebit TrailBlazer

The TrailBlazer needs to have internal registers set in order to use it properly with `tip` and `uucp`. The first time you are connected to the modem, the following setup sequence need to be entered to setup the TrailBlazer for use:

```
AT &F Q6 S51=254 S52=2 S53=1 S54=3 S58=0 S111=30 &W
```

You can set this with `tip` by entering

```
% tip cua0
```

(if there is an entry for `cua0` in `/etc/remote`) and entering the sequence after you get the message informing you that you are connected. This is a one time operation. Note that the above sequence may be slightly different from the one shown in the TrailBlazer documentation. Sun has found that the above sequence works well for TrailBlazers connected to Sun systems, but you need read the TrailBlazer documentation fully to make sure it is right for you.

Settings for other modems

Hayes compatible modems that use the Hayes `AT` command set *may* work with Sun `tip` and `uucp` software. Configure the modem as follows:

- Hardware DTR, that is, when the Sun drops DTR (as when someone logs off) the modem should hang up.
- Hardware Carrier Detect, that is, the modem only raises the Carrier Detect (CD) line when there is an active carrier signal on the phone connection, when carrier drops, either when the other end of the connection terminated or in the event the phone connection is broken, the Sun will be notified and act appropriately. The CD signal is also used for coordinating dial-in *and* dial-out use on a single serial port and modem.
- Respond with numeric result codes.
- Sends result codes.
- Does not echo commands.

Kernel Modification

A new feature in SunOS 4.1 is the `ttysoftcar` command. This command allows you to set the software carrier detect of devices while a machine is running, so that it is no longer necessary to modify the kernel when adding a modem to your system. If you are adding a modem to a communications board that is not supplied by Sun, there is a chance that the hardware will not support the `ttysoftcar` command. If you know for a fact that your hardware doesn't support this feature, skip to the section titled *Kernel Modification for Boards Not Supporting ttysoftcar*.

System Modification

You have to run various commands and edit the `/etc/ttytab` file to implement the support services that your machine will require. To allow for terminal and full modem support, you will need to run `mknod` and edit the `ttytab` files as shown in the following sections. You must be superuser to make the following system modifications.

Using `mknod`

First create an alternate device for your modem in the `/dev` directory by using `mknod`. This alternate device allows a single tty line to be connected to a modem and used for both incoming and outgoing calls. If you do not wish to allow for outgoing calls, skip to the next step. The syntax for `mknod` is as follows:

```
mknod filename [c]|[b] major_device_number minor_device_number
```

There are several arguments which you must give to `mknod`. The first is the name of the device you are making. It will be `cua*`, where the asterisk is the logical value of this alternate device on your system — 0 for the first alternate device, 1 for the second and so forth. The second argument will always be `c`, indicating that the device is a character (or *raw*) device. A raw device deals with each character individually rather than working with blocks of characters. The last two arguments are the major and minor device numbers. To determine what numbers to use here do an `ls -l` on the `/dev/tty*` device you are making the alternate device for. If you are going to use the first port on your CPU board, `ttya`, do the following:

```
# ls -l /dev/ttya
crw--w--w- 1 root      12,   0 Sep 17 18:27 /dev/ttya
```

Here the major device number is 12 and the minor device number is 0. When running `mknod` to create an alternate device for a modem, give the actual major device number and you always add 128 to the minor device number. Thus, in this example, the `mknod` command takes 12 and 128 as its last two arguments. You also change the mode and ownership of this device. The whole process would look like the following:

```
# cd /dev
# mknod cua0 c 12 128
# chmod 600 cua0
# chown uucp cua0
```

In this example, an alternate device `cua0`, the special file for the first modem attached to a system, was created. The second modem attached would be `cua1`, and so forth.

Updating ttytab

Assuming there is an entry in `/etc/gettytab` like the following, for a modem suitable for use at both 1200 and 300 baud:

```
# fast dialup terminals, 1200/300 rotary (can start either way)
#
3|D1200|Fast-Dial-1200:\
  :nx=D300:fd@:tc=1200-baud:
5|D300|Fast-Dial-300:\
  :nx=D1200:tc=300-baud:
```

Edit the `/etc/ttytab` so it will look like the following for both dialin and dialout capability, connected to `ttya`:

Figure 11-2 `tttytab` for systems with boards supporting `tttysoftcar`

```
#
# name getty                type                status  comments
#
console "/usr/etc/getty std.9600"    sun                on local secure
ttya    "/usr/etc/getty D1200"       dialup             on remote
ttyb    "/usr/etc/getty std.9600"    unknown            off local secure
.
.
.
```

In the example above, the status of the dialup device is "on", indicating that you want dialin capability. If you do not want dialin capability, it should be "off" as shown here:

```
ttya    "/usr/etc/getty D1200"       dialup             off remote
.
.
.
cua0    "/usr/etc/getty D1200"       unknown            off remote secure
```

If you want to use a modem at a single baud rate only, set up `/etc/ttytab` as you would a directly connected terminal. For example, the following `gettytab` entry

```
f|std.1200|1200-baud:\
  :fd#1:sp#1200:
```

can be used for a fixed 1200-baud line. Modify `/etc/ttytab` accordingly:

```

#
# name  getty                type          status  comments
#
console  "/usr/etc/getty std.9600"  sun          on local secure
ttya     "/usr/etc/getty std.1200"    dialup       on remote
ttyb     "/usr/etc/getty std.9600"    unknown      off local secure
.
.
.
cua0     "/usr/etc/getty D1200"       unknown      off remote secure

```

In the above examples, the status field “local” has been replaced with “remote” to indicate that the device is a modem. The actual field entry is not important, as anything other than “local” will work. The term “remote” is used merely as the logical converse of “local.” Do not remove the `ttyp*` or `ttyq*` entries in the `/etc/ttytab` file; they are required for the window system and other programs.

After you have finished editing `/etc/ttytab`, you must reinitialize the file by notifying `init` with the `kill` command:

```
# kill -1 1
```

The `/etc/ttytype` file is automatically updated to reflect information in `/etc/ttytab`.

Now, make sure that the following line in the script `/etc/rc` is not commented out with a pound sign “#”:

```
/usr/etc/ttysoftcar -a > /dev/null 2>&1
```

This will insure that the software carrier detect is turned off for all devices listed in `/etc/ttytab` that are marked as “remote” every time you reboot. If you do not wish to reboot at this time, you should also enter the command manually, as shown here:

```
# ttysoftcar -a
```

Kernel Modification for Boards Not Supporting `ttysoftcar`

The following is the “hard way” of informing your system of the new modem and should only be necessary in the case of adding a modem to a non-Sun supplied board. “Third-party” boards may not support the `ttysoftcar` command, which sets the software carrier detect signal without needing to actually modify the kernel. If you have followed the procedures dealing with `ttysoftcar` as shown above, skip to the section *The /etc/remote File*.

You must be superuser to make the following kernel modifications.

You must edit the device driver specification for the serial communications controller in your configuration file, to enable carrier detect on the line where the

modem is attached. This is referred to as turning off the software carrier, which then allows the system to detect a hardware carrier on the line. When turned on the system treats each line as hard-wired with carrier always present. Turning on or off is done by specifying a parameter in the `flags` field. For example, below is a sample entry from a kernel configuration file for the serial communications controller on the standard Sun-3 CPU board:

```
device zs0 at obio ? csr 0x20000 flags 3 priority 3
```

To make the device in this example into a modem-compatible driver, you have to change the hexadecimal value `0x3` after `flags`. The same field will have to be changed for other device drivers if your modem is going to be attached to a board other than the Sun CPU board.

Specifically, you need to make the following changes to the device driver specification line in the kernel configuration file. The changes for each of the three boards to which a modem can be attached are given below.

The Standard Sun CPU Board — The serial communications controller on the standard Sun CPU board, device `zs0`, provides two lines with full modem control. For both dial-in and dial-out on the same serial port, the `flags` bit in the kernel corresponding to that serial port has to be set to zero. This enables hardware carrier detect so that the Sun machine can tell when someone dials in or hangs up. The default value of `flags` for `zs0` in the GENERIC kernel is 3, indicating software carrier for both ports a and b. To permit hardware carrier detect on `ttya`, `flags` should be changed to `0x2`, on `ttyb` to `0x1`, and on both to `0x0`.

A Systech MTI Board — The Systech MTI-800 and MTI-1600 boards require a kernel configuration line for device `mti0`. As above, the `flags` value is set in hexadecimal. The default `flags` value on installation of either MTI board is `0xffff`. This value selects software carrier detect for all lines. For lines with modems, you need hardware carrier detect and must set the corresponding `flags` bit in the kernel to zero, just as in the examples above.

An ALM-2 Board — The Sun ALM-2 board require that a kernel configuration line for device `mcp0`. As always, the `flags` value is set in hexadecimal notation (the default is `0x1ffff`.) To set a hardware carrier detect, the corresponding `flags` bit must be set to zero.

The following example shows how to determine the proper hexadecimal values for the `flag` bits on a Systech MTI-1600 board where ports zero through 7 are turned on, software carrier detect, and ports 8 through f are turned off, hardware carrier detect for modem operation. The method shown here can be applied to all serial ports.

Shown below is what can be thought of as the “switch” settings for each of the available ports. If the “switch” is on, it is up, and if it is off, it is down:

port	f	e	d	c	b	a	9	8	7	6	5	4	3	2	1	0
on									1	1	1	1	1	1	1	1
off	0	0	0	0	0	0	0	0								

If you take the values shown here, '0000' '0000' '1111' '1111', assume that they are binary numbers, and convert to hexadecimal, you arrive at the entry for the `flags` parameter: `0x00ff`. An easy method to do this for any given binary number to hexadecimal is as follows:

```
% echo "16o2inumberp" | dc
```

Where *number* is replaced with the binary number to be converted to hexadecimal.

For more information on the serial communications driver and the multi-terminal interface, see the pages `zs(4s)` and `mti(4s)` in the *SunOS Reference Manual*.

Here are the steps to edit and install the new kernel on a system named `CHAOS`. (The following example configures a kernel for a Sun-3 to use a modem on the `ttya` serial port.)

1. Go to `/usr/share/sys/sun[3,3x,4,4c]/conf` directory and make a copy of the current kernel configuration file to keep until the new one is installed and running.

```
# cd /usr/share/sys/sun[3,3x,4,4c]/conf
# cp CHAOS old.CHAOS
```

If you have not built your own kernel, you have to base your kernel configuration on the `GENERIC` kernel configuration file as shown below:

```
# cd /usr/share/sys/sun[3,3x,4,4c]/conf
# cp GENERIC CHAOS
# chmod +w CHAOS
```

2. Edit the kernel configuration file, changing the `flags` bit, as explained above, for the board to which you will attach the modem. For example, if you want to put the modem on `ttya`, and your entry looks like this:

```
device zs0 at obio ? csr 0x20000 flags 3 priority 3
```

You need to change it to look like this:

```
device zs0 at obio ? csr 0x20000 flags 0x2 priority 3
```

3. Run `/usr/etc/config` on the new kernel configuration file.

```
# /usr/etc/config CHAOS
```

4. Build the new kernel.

```
# cd ../CHAOS
# make
```

5. Now install the new kernel and try it out.

First move the original working kernel to another (safe) place. Then copy the new kernel to the place of the original. Finally boot up the system with this new kernel.

```
# mv /vmunix /vmunix.old
# mv vmunix /vmunix
# /etc/halt
```

The system goes through the halt sequence. Then the monitor displays its prompt, at which point you can boot the system by typing **b**.

The system boots up multiuser, and then you can resume normal operations (if all is well.)

6. If the new kernel does not seem to function properly, halt the system and boot from the original kernel. Then reinstall the original kernel. Once you are booted up on the original, you can go about trying to fix the faulty kernel.

Remember to remove the `old.CHAOS` kernel configuration file you created as a backup.

For more information about kernel building, see *Reconfiguring the System Kernel* and *Advanced Administration Topics*.

System Modification

You have to run various commands and edit files to implement the support services that your machine will require. To allow for terminal and full modem support, you will need to run `mknod` and edit the `ttytab` files as shown in the following sections. You must be superuser to make the following system modifications.

Using `mknod`

First create an alternate device for your modem in the `/dev` directory by using `mknod`. This alternate device allows a single tty line to be connected to a modem and used for both incoming and outgoing calls. If you do not wish to allow for outgoing calls, skip to the next step. The syntax for `mknod` is as follows:

```
mknod filename [c]|[b] major_device_number minor_device_number
```

There are several arguments which you must give to `mknod`. The first is the name of the device you are making. It will be `cua*`, where the asterisk is the logical value of this alternate device on your system — 0 for the first alternate

device, 1 for the second and so forth. The second argument will always be `c`, indicating that the device is a character (or *raw*) device. A raw device deals with each character individually rather than working with blocks of characters. The last two arguments are the major and minor device numbers. To determine what numbers to use here do an `ls -l` on the `/dev/tty*` device you are making the alternate device for. If you are going to use the first port on your CPU board, `ttya` do the following:

```
# ls -l /dev/ttya
crw--w--w-  1 root          12,   0 Sep 17 18:27 /dev/ttya
```

Here the major device number is 12 and the minor device number is 0. When running `mknod` to create an alternate device for a modem, give the actual major device number and you always add 128 to the minor device number. Thus, in this example, the `mknod` command takes 12 and 128 as its last two arguments. You also change the mode and ownership of this device. The whole process would look like the following:

```
# cd /dev
# mknod cua0 c 12 128
# chmod 600 cua0
# chown uucp cua0
```

In this example, an alternate device `cua0`, the special file for the first modem attached to a system, was created. The second modem attached would be `cua1`, and so forth.

Now move the `tty` device whose software carrier you turned off in the `flags` field of your device driver specification in the kernel above, to a dialing device. This is the same one for which you created the alternate device with `mknod`. In this example you move the first port on the CPU board to the first modem device.

```
# mv ttya ttyd0
```

The second modem device would be `ttyd1`, and so forth.

Updating `ttytab`

Now edit the `/etc/ttytab` file. First, add an entry at the end for the new `ttyd0` line, and disable logins for the no longer existing `ttya` device.

A modem suitable for use at both 1200 and 300 baud could use a `gettytab` entry similar to the following:

```
# fast dialup terminals, 1200/300 rotary (can start either way)
#
3|D1200|Fast-Dial-1200:\
    :nx=D300:fd@:tc=1200-baud:
5|D300|Fast-Dial-300:\
    :nx=D1200:tc=300-baud:
```

The resulting `/etc/ttytab` file will look like the following:

Figure 11-3 *ttytab for systems with boards not supporting ttysoftcar*

```

#
# name getty                type                status  comments
#
console "/usr/etc/getty std.9600"    sun                on local secure
ttya    "/usr/etc/getty std.9600"    925                off local secure
ttyb    "/usr/etc/getty std.9600"    unknown            off local secure
.
.
.
ttyd0   "/usr/etc/getty D1200"          dialup             on remote
cua0    "/usr/etc/getty D1200"          unknown            off remote secure

```

In the examples above, the status of the dialup device is “on”, indicating that you want dialin capabilities. If you do not wish it to be possible to dialin to this modem, you should turn the device off by changing the status to “off.” Notice that the `ttya` line has the status disabled, while the `ttyd0` line has dialins enabled at either 1200 or 300 baud — sending a break through the modem causes `getty` to switch to the other baud rate. If you want to use a modem at a single baud rate only, set up `/etc/ttytab` as you would a directly connected terminal. For example, you can use the following `gettytab` entry

```
f|std.1200|1200-baud:\
:fd#1:sp#1200:
```

can be used for a fixed 1200-baud line. Modify `/etc/ttytab` accordingly:

```

#
# name getty                type                status  comments
#
console "/usr/etc/getty std.9600"    sun                on local secure
ttya    "/usr/etc/getty std.1200"    dialup             on remote secure
ttyb    "/usr/etc/getty std.9600"    unknown            off local secure
.
.
.
cua0    "/usr/etc/getty D1200"          unknown            off remote secure

```

In the examples shown above, the status field “local” has been replaced with “remote” to indicate that the device is a modem. The actual field entry is not important, as anything other than “local” will work. The term “remote” is used merely as the logical converse of “local.” Do not remove the `ttyp*` or `ttyq*` entries in the `/etc/ttytab` file; they are required for the window system and other programs.

After you have finished editing `/etc/ttytab`, you must reinitialize the file by notifying `init` with the `kill` command:

```
# kill -1 1
```


The `/etc/ttytype` file is automatically updated to reflect information in `/etc/ttytab`.

The `/etc/remote` File

The file `/etc/remote` stores the attributes of systems that you dial out to using `tip(1)`, as explained shortly. It is structured somewhat like `termcap`. If you wish to have `tip` dialout capability, you must now edit that file. Each line in the file provides a description for a single known system. Fields are separated by a colon (:). Lines ending in a backslash (\) and followed immediately by a newline (carriage return) are continued on the next line. Look in the `/etc/remote` file or see the `remote(5)` manual page in the *SunOS Reference Manual* for further explanation. If you want to connect to the modem directly, make sure the following entry exists

```
cua0:dv=/dev/cua0:br=#1200
```

The entry shown is for a 1200 baud modem on `/dev/cua0`. If you are using a modem at a different baud rate, you use that baud rate instead of 1200.

The following example is for connecting to a system named “sunphone,” at phone number 7630927, at baud-rate 1200, using the Hayes auto call unit protocol, on dialer `cua0`:

```
sunphone:\
:pn=7630927%:tc=UNIX-1200:
UNIX-1200:\
:el="D^U^C^S^Q^O@:du:at=hayes:ie=#$:oe="D:br#1200:tc=dialers:
dialers:\
:dv=/dev/cua0:
```

For dialout capability with `cu`, see *Administering the UUCP System*

Hayes Specific Considerations

tip Support — To use a Hayes modem with `tip`, you should specify the modem type in the `/etc/remote` file as either `at=hayes` or `at=ventel`. The phone number attribute (`pn`) can contain any valid dial commands; see your modem manual for details. The most common commands to the phone number attribute are:

- A phone number of numeric characters 0-9.
- A comma (,) causes a 2 second pause to wait for a secondary dial tone. For example to dial an outside line from a local phone network.
- A P or T switches to pulse or tone dialing. By default `tip` uses tone dialing.
- You may set parameters for the dialer by starting the phone number with an “S” (for set) flag. Read the Hayes manual for an explanation.
- Note that `tip` can cycle through a set of phone numbers, dialing each one until a connection is made. Phone numbers must be separated with a vertical bar (|), just like the separator in the name field. For example, the `/etc/remote` line for `pn` might look something like:

```
:pn=2138896565|2138896564|2138896563:
```

For more information regarding the use of `tip`, see the *SunOS User's Guide: Getting Started*.

Problems

Even the most routine of installation can sometimes hit unexpected snags. If something does not appear to be working as it should after following the procedures given above, the following frequent corrections can be tried

- First, check to make sure all devices (modems, terminals, and computer) are properly turned on. It is surprising how often even the most experienced System Administrators make this mistake.
- If you still have problems after installation, check the cabling: as outlined in the section *Connecting Devices to Asynchronous Serial Ports*, and make sure you have pins 2 through 8 and 20 wired straight through. (If MTI, wire 2, 3, 7, and 20 straight through and have pin 8 on the modem go to pin 6 on the MTI port also you need to tie 4 to 5 on the MTI side.) Next, check the information in the `/etc/ttytab` and the `/etc/remote` files, and make sure you have restarted `init` as explained.
- If you cannot access a port, and find a process running on it when you do a `ps -ax`, then make sure you have the 8 pin connected in your cable. If that does not work, check to make sure your device driver is configured properly to set the correct flag for the line to `off`.
- Sometimes even when both the hardware and software are correct, the device driver gets into a state where it will not let the alternate port be opened. You must do a `kill -1 1` to notify `init` and reset the flags on the device driver.
- If you get a

```
can't synchronize with hayes
```

error message when using a Hayes-compatible modem, check internal and external modem switch settings, and check the cable connection. Power cycle the modem if necessary.

- If you get a

```
can't synchronize with ventel
```

error message when using a Hayes-compatible modem, look in the `/etc/remote` file and make sure you have changed `at=ventel` to `at=hayes`.

- The message

```
tip: /dev/cua0: No such file or directory
```

usually means that the device special file is missing from `/dev` (or is incorrect there).

- The message

```
tip: /dev/cua0: No such device or address
```

means that device driver is missing from your kernel.

- The message

```
all ports busy
```

may mean that the port is actually busy running `tip` or a dial-in user. You can do a `ps -ax` to see what is running.

You should check to make sure that you have set up the carrier detect properly as well. In SunOS 4.0 and later, `getty` runs at all times (blocking when there is no dialin), so the mere presence of a `getty` is not a problem indicator as in previous releases.

Check:

- the flags bit in the kernel
- the cable the modem settings
- the `/etc/ttytype` file for correct device entry.

If no process is currently using the serial port, there may be a leftover lock file. Look for a lock file in `/var/spool` and `/var/spool/locks` and remove it. It will have a name like `LCK.cua0`. If you change any modem switch, unplug and power cycle the modem as explained above. If you get the above message when `tip` is not running, no one is dialed in, and there is no lock file, try unplugging the modem cable and/or power cycling the modem. Finally, you can do `ps -ax` and look for a process tying up the port.

- If you get a

```
tip: /dev/cua0: Permission denied
```

or

```
link down
```

error message, make sure you have `:dv=/dev/cua0: in /etc/remote`. Check for a lock file in `/var/spool` or `/var/spool/locks` called `LCK.*` where `*` is the name of the dial out destination. Make sure permission modes on `/dev/cua0` are `600`, and it is owned by `uucp`. Try turning off the modem, unplugging it for a minute, and plugging it back in again. Also check permissions on the `/var/spool/locks` directory. It must exist and be readable, writable, and executable by everybody (`777` mode).

11.5. Adding a Printer to Your System

The SunOS operating system can handle a variety of printers and configurations. It handles local and remote printers, printers attached to serial lines, raster output devices such as Varian or Versatec, and laser printers such as an Imagen or the Sun LaserWriter.

Some printers require customer software for handling communication between a Sun workstation and printer or for converting data into an acceptable form required by the printer. The Imagen and Sun LaserWriter are examples of such printers. Contact the printer vendor for information on this software; for example, contact Sun for the Transcript software that provides the necessary support for printing on the Sun LaserWriter. For the software installation procedures, refer to the chapter *Maintaining Printers and Print Servers*.

Serial Port and Cable Connection

This section deals with the hardware configuration of the printer. Read the manufacturer's printer manual and check the following on the new printer.

- Determine whether the printer is configured as DTE (data terminal equipment) or DCE (data communication equipment). Set it to DTE if you can.
- If possible, disable the printer's use of modem control signals like CTS (clear-to-send) and DSR (data-set-ready). If the printer requires CTS and DSR, loop back the following lines on the printer: connect a loop of wire from line 4 to line 5, and from line 6 to line 20.
- If possible, enable xon/xoff flow control.
- At least for the installation phase, set the baud rate at a low rate for testing, 1200 for example.

Now make sure you have an available serial port on your workstation. When you do have a port available, connect the printer to the workstation with a three-line cable. If the printer is a DTE device, use a null-modem cable. Null modem cables have line 7 wired straight through and lines 2 and 3 swapped so that proper transmit and receive signals are communicated between two DTE devices. If the printer is a DCE device, then connect 2, 3, and 7 straight through. Make sure all the connections are tight.

When you have connected the printer as explained above, you should verify that cabling and hardware are performing before proceeding. Obviously, if there is a problem with connection or faulty hardware, none of the later software installation will work.

- Consult the operation manual for the printer and run the printer's standalone diagnostics. If these do not run, call the printer manufacturer. Remember to set the switch on the printer back to operate mode after the self-test.
- Verify that the printer switch settings are correct. Check for the correct settings in the printer operation manual, and make sure they correspond with the parameters given above for the Sun.
- Send something over the tty line to the printer. To do this, set the characteristics of the printer using `stty`, and then use `cat` to send a file to the `/dev/tty*` serial port where the printer is attached. If the printer is

attached to the first serial port on the CPU board, type:

```
# (stty 1200; cat /etc/passwd) > /dev/ttya
```

to print a copy of the password file.

If the printer is “dead,” your cable may be bad, or you may need a null modem cable, as described above.

If you get garbled printouts, `stty` may not have set all the terminal characteristics properly for the printer. Read the manufacturer’s manual and check the switches on the printer. In particular check the baud rate, 1200 in this example. Make sure the printer and the `stty` setting match. You may have to set additional options with `stty` to match those on your printer. The most likely are:

- to set parity to `evenp`, `oddp` or `neither` (omit parity with `-parity`); to set tab expansion with `-tabs`
- Allow carriage return for newline, and output CR-LF for carriage return or newline with `-onlcr`.
- These parameters are set for the line printer spooling system software in the `/etc/printcap` file by the `ms` capability. See below for an explanation. Also see `printcap(5)` in the *SunOS Reference Manual*.

File Modifications

Once the printer is physically attached, a few file modifications must be made.

- Make sure that `init(8)` does not create a login process on the port you are using for your printer. To do this, edit the `/etc/ttytab` file, making sure that the “status” field is “off” for the `tty` port that your printer is attached to. The example `printcap` attaches a serial printer to the first CPU serial port, `ttya`, `/etc/ttytab` should have an entry like:

```
ttya    "/usr/etc/getty std.9600"    925 off
```

If it was necessary to edit the `/etc/ttytab` file, you must notify `init` to make the system aware of the changes made. Issue the following `kill` command while you are superuser:

```
# kill -HUP 1
```

Now you can send output to the printer with the `lpr` command. If the printer does not work now, but did when you sent it output with the `cat` command to `/dev/tty*`, make sure the permissions and characteristics of all the `lpr`-related files are correct. They must be as described in the chapter *Maintaining Printers and Print Servers*. Make sure that your output filter (if you have one) is executable and that it is located where `/etc/printcap` says it is. Finally, check all the fields and the syntax in the `/etc/printcap` file.

Hooking Up a Printer to a VPC-2200 Multibus Board

In this configuration, you hook-up your printer to a Multibus printer controller. Sun currently provides support in the standard software distribution for a single Systech VPC-2200 board per workstation. This section explains the installation of that board and a printer.

If you want to use a different printer controller board you will have to write your own device driver for the kernel. This requires some expertise; the procedure is described in *Writing Device Drivers*. Due to the difficulty of writing a driver, this course of action is not recommended.

The remaining discussion deals with the Systech VPC-2200 board.

First install the board in the card cage of your Sun workstation and connect the cable. If necessary, configure the new board into the system kernel. Finally modify the necessary files to enable the workstation to queue jobs and send them to the printer.

These steps are discussed in the sections below.

Card Cage Installation And Cable Hook-up

Read the manufacturer's manual for the VPC-2200. Check all recommendations and settings carefully. When installing a printer controller board, you may place it in any slot that does not share a P2 section with the Sun-3 CPU or Sun-3 Memory boards. A basic ribbon cable can connect the board to the printer.

Follow the self test instructions for the board once it is installed. (Note that some printers, such as the Imagen, will not work with self test.) Remember to set switches back to operate mode after self test is complete. If you have trouble with self test, double check all the recommended switch settings, check the cabling from the controller to the printer, and make sure the board is connected snugly in the card cage. Consult the manufacturer's manual for any recommended procedures. Call the manufacturer if the board still seems flaky.

Kernel Modification

You must be superuser to make kernel modifications.

Here is a brief outline of the steps for building a new kernel after installing a Systech VPC-2200 printer interface board.

Follow the steps shown here to reconfigure the kernel after the installation of a Systech printer controller board. These steps explain the process on a system named GRENDL:

1. Go to `/usr/share/sys/sun[3,3x,4,4c]/conf`, and make a copy of the current kernel configuration file to keep until the new one is installed and running.

```
# cd /usr/share/sys/sun[3,3x,4,4c]/conf
# cp GRENDL old.GRENDL
```

If you have not built your own kernel, you will have to base your kernel configuration on the GENERIC kernel configuration file. Make preparations for building your new kernel as follows:

```
# cd /usr/share/sys/sun[3,3x,4,4c]/conf
# cp GENERIC GRENDEL
# chmod +w GRENDEL
```

2. Edit the kernel configuration file, adding an entry for the Systech board you have installed into the card cage. Look in *Section 4* of the *SunOS Reference Manual* for a description of this device. The entry you make in the kernel configuration file looks like this:

```
device vpc0 at vme16dl6 ? csr 0x480 priority 2 vector vpcintr 0x80
```

3. Run `/usr/etc/config` on the new kernel configuration file.

```
# /usr/etc/config GRENDEL
```

4. Build the new kernel.

```
# cd ../GRENDEL
# make
```

5. Now install the new kernel and try it out.

Move the original working kernel to another (safe) place. Copy the new kernel to the place of the original. Finally boot up the system with this new kernel.

```
# mv /vmunix /vmunix.old
# mv vmunix /vmunix
# /etc/halt
```

The system goes through the halt sequence. Then the monitor displays its prompt, at which point you can boot the system by typing **b**.

The system boots up multiuser, and then you can resume operations.

6. If the new kernel does not seem to function properly, halt the system and boot from the original kernel. To do this, issue the following command at the monitor prompt:

```
> b /vmunix.old
```

Then reinstall the original kernel. Once you have booted the original kernel, you can fix the faulty kernel.

Remember to remove the `old.GRENDEL` kernel configuration file you created as a backup.

For more information about kernel building, see *Reconfiguring the System Kernel* and *Advanced Administration Topics*.

File Modification

You must be superuser to make the following file modifications.

Using MAKEDEV To Create Special Files

The kernel communicates with the printer through special files in the directory /dev. When a Systech board is added to the system, these new entries must be made in the /dev directory. This is relatively easy to do with MAKEDEV. MAKEDEV takes the argument vpc0, and automatically installs the files /dev/vpc0 and /dev/lp0 (vpc0 is the special device file for the Versatec interface; lp0 is the special device file for the Centronics/Dataproducts interface.)

```
# cd /dev
# MAKEDEV vpc0
```

If the system had a printer in the past, there may be a /dev/vpc0, or even a /dev/lp0, existing already. If they exist, you **should** remove these files before running MAKEDEV. An error message from MAKEDEV,

```
mknod: File exists
```

will be returned if you try to make a file on an existing one.

Editing the /etc/printcap File

Refer to the chapter *Maintaining Printers and Print Servers* for this information.

Maintaining Printers and Print Servers

This chapter deals with setting up and maintaining the SunOS line printer spooling system. To actually install the printer hardware, read the chapter *Adding Hardware to Your System*. Follow the procedures outlined in that chapter for adding a printer to your system. Once the hardware is properly installed, you should read this chapter to get the printer software running properly.

12.1. Introduction

A printer spooling system is a suite of programs that allow a computer to “spool” or queue up printer requests and then route them to the proper printers. Using the SunOS line printer spooling system, it is possible to have many users sending print jobs to printers anywhere on the network. To introduce you to the line printer system, the following sections explain what the capabilities and duties of the various components are. Following those explanations, the proper procedures for installing and maintaining those components are given.

12.2. Overview

The line printer system supports:

- Multiple printers
- Multiple spooling queues
- Both local and remote printers
- Printers attached via serial lines that require line initialization such as the baud rate
- Raster output devices such as a Varian or Versatec
- Laser printers such as a LaserWriter or an Imagen

The line printer system consists mainly of the following files and commands:

<code>/etc/printcap</code>	printer configuration and capability database
<code>/usr/lib/lpd</code>	line printer daemon, does all the real work
<code>/usr/ucb/lpr</code>	program to enter a job in a printer queue
<code>/usr/ucb/lpq</code>	spooling queue examination program
<code>/usr/ucb/lprm</code>	program to delete jobs from a queue
<code>/etc/lpc</code>	program to administer printers and spooling queues
<code>/dev/printer</code>	socket on which lpd listens

The file `/etc/printcap` is a master database describing line printers directly attached to a machine and, also, printers accessible across a network. The manual

page entry `printcap(5)` provides the authoritative definition of the format of this database, as well as specifying default values for important items such as the directory in which spooling is performed.

12.3. Commands

The following is a description of the commands related to the printer spooler system. They will be discussed in depth later on in the chapter, once their use has become clear.

The `lpd` daemon acts as a master server for coordinating and controlling the spooling queues configured in the `/etc/printcap` file. It is usually invoked at boot time from the `/etc/rc` file. When `lpd` is started it makes a single pass through the `printcap` database restarting any printers that have jobs.

Use the `lpr` command to enter a print job in a local queue and to notify the local `lpd` that there are new jobs in the spooling area. `lpd` either schedules the job to be printed locally, or if printing remotely, attempts to forward the job to the appropriate machine. If the printer cannot be opened or the destination machine is unreachable, the job will remain queued until it is possible to complete the work.

The `lpq` program works recursively backwards displaying the queue of the machine with the printer and then the queue(s) of the machine(s) that lead to it. `lpq` has two forms of output: in the default, short, format it gives a single line of output per queued job; in the long format it shows the list of files, and their sizes, that comprise a job.

The `lprm` command deletes jobs from a spooling queue. When removing jobs destined for a remote printer, `lprm` acts similarly to `lpq` except it first checks locally for jobs to remove and then tries to remove files in queues off-machine.

You can use the `lpc` program to alter the operation of the line printer system. For each line printer configured in `/etc/printcap`, use `lpc` to:

- Disable or enable a printer
- Disable or enable a printer's spooling queue
- Rearrange the order of jobs in a spooling queue
- Restart the `lpd` daemon
- Find the status of printers and their associated spooling queues and printer daemons. This command is explained further in the section *Line Printer Administration* later in this chapter.

12.4. Setting Up

SunOS Release 4.1 comes with the necessary programs installed and with the default line printer queue created.

The real work in setting up is to create the `printcap` file and any printer filters for printers not supported in the distribution system.

The /etc/printcap File

You need to create a /etc/printcap file to make the printer work properly. The printcap database describes line printers directly attached to a machine and printers accessible across a network. Each printer should have a single entry. It is a good idea to remove entries for printers not in your system. Typically, a system will have just a single printer. The syntax of entries in printcap can seem tortuous; check carefully after you have modified the file. For a full explanation of printcap, see the man pages.

Editing the printcap File

printcap entries consist of fields that (except for the name field) *must* be preceded and followed by colons (:); the last field specified must terminate with a colon. The first field of each entry *must* be the name(s) the printer is known by, where these names are separated by 'l' characters, and the field begins at the left margin without a leading colon. Every system *must* have one, and only one printer that uses lp as one of its aliases. In a single printer environment, you must include lp in the name field. Here is a sample printcap entry for a printer attached to a serial port.

```
# printcap entry for printer on serial port
myprinter|lp|ps|lw|Print_Server:\
:lp=/dev/ttya:br#9600:\
:ms=-parity,onlcr,ixon,decctlq:\
:sd=/var/spool/myprinter:\
:lf=/var/adm/lpd-errors:
```

You can learn some general information from this example. First, notice that comments are allowed if a line begins with a '#'. Next something not so obvious to see, an entry is continued onto another line with the '\n' character followed, without blank space, by a carriage return. This is a *must*. Blank space accidentally typed at the end of a printcap line will cause severe problems. Notice that the continuation lines of this example are tabbed over from the left margin; all continuation lines in an entry *must* begin with blank space, normally a tab character. When entries do continue onto second and subsequent lines, a colon *must* appear at the end of one line and the beginning of the next.

Now, consider the fields within each entry. The printcap(5) manual page gives a table of all the capabilities available. You will probably need just the ones shown here to run most line printers from a serial port.

As mentioned above, the first field of each entry gives the names the printer is known by. One of the names must be lp; on a machine with more than one printer, one printer must use lp as one of its names — but only one printer. If lp is not assigned to any printer, commands such as lpr and lpq will not be able to determine which printer to use as a default.

Notice that each subsequent field is introduced by a two character code. Numeric capabilities take the form: *character_code#number_value*; for example, br#9600. String capabilities take the form: *character_code=sequence*; for example, lf=/var/adm/lpd-errors. The capabilities shown in this entry are:

lp	Specifies the file name to be opened for output. In this case it is the first serial port <code>/dev/ttya</code> . The default is <code>/dev/lp</code> .
br	Sets the baud rate for the tty line to the value given here.
ms	Sets the printer characteristics. In this example, there is no parity, newlines are printed as carriage return/line feed pairs, XON/XOFF flow control is used, and XON controls are accepted only after an XOFF. See <code>stty(1)</code> for more information on the <code>ms</code> options.
sd	Specifies the name of a spooling directory. Make sure the directory exists with proper permissions before attempting to run the printer. When you install the operating system, it includes a correct spool directory, which is <code>/var/spool/lpd</code> . You can change this to whatever you like if you intend to have your machine connected to more than one printer. In this example, the spool directory is <code>/var/spool/myprinter</code> .
lf	This is where spooler errors are logged. It can be created anywhere, but must exist with write permissions before errors can be sent to it. To assure that the spooler daemon can write on the error log file, become superuser, create the log file, and do a <code>chmod 666</code> on it.

Printers on Serial Lines

When a printer is connected via a serial communication line it must have the proper baud rate and terminal modes set. The following example is for a printer connected locally via a 9600 baud serial line.

```
lp|printer|localprinter:\
:lp=/dev/lp:br#9600:\
:ms=-parity,onlcr,crtsets:
:of=/usr/local/printtools/printof:lf=/var/adm/lpd-errs:
```

The `lp` entry specifies the file name to open for output. Here it could be left out since `/dev/lp` is the default. The `br` entry sets the baud rate for the tty line. The `ms` entry indicates that there is no parity that the printer expects, that newlines are to be carriage return/line feed pairs, and that hardware RTS/CTS flow control is to be used. The `of` entry specifies that the filter program *printof*, in the `usr/local` directory should be used for printing the files; more will be said about filters later. The last entry causes errors to be written to the file `/var/adm/lpd-errs` instead of the console. Most errors from `lpd` are logged using `syslogd(8)` and will not be logged in the specified file. Only those that write to standard error output will end up with errors in the `lf` file.

Printers on Parallel Ports

Some Sun systems provide a parallel port that can be used to drive a suitable printer. These systems include the Sun 3/80 and systems including a Sun ALM-2 board.

The `/etc/printcap` entry is simpler than for serial lines as there is no baud rate or terminal modes to worry about. Here is a sample `/etc/printcap` entry for a printer on such a port

```
lp|parallel printer|printer:\
:lp=/dev/mcpp0:sd=/var/spool/lpd:
```

The device name `/dev/mcpp0` is the parallel port on the first ALM-2 board. If you are connecting to a Sun 3/80, the device you would use would be `/dev/pp0`.

Like any other printer, you may specify filters and log files as you desire, using the standard syntax.

Remote Printers

Printers that reside on remote hosts should have an empty `lp` entry. For example, the following `printcap` entry would send output to the printer named “lp” on the machine `einstein`.

```
lp|default line printer:\
:lp=:rm=einstein:rp=lp:sd=/var/spool/vaxlpd:
```

The `rm` entry is the name of the remote machine to connect to; this name must be a known host name for a machine on the network. The `rp` capability indicates the name of the printer on the remote machine is “lp”. Here it could be left out since this is the default value. The `sd` entry specifies `/var/spool/vaxlpd` as the spooling directory instead of the default value of `/var/spool/lpd`.

Other File Modifications

After your `printcap` file has been edited for your printer(s), modify a few more files.

- Check to make sure the proper permissions and ownerships exist on the files `/usr/lib/lpd`, `/usr/ucb/lpr`, and on the directory, `/var/spool/lpd`. Note that you may have given a different name to your `/var/spool/lpd` directory, and that you will have one spooling directory with a unique name for each printer accessible on your system. For the files named above type `ls -lg` to check permissions, ownership and group. To check the same things on the spooling directory, type `ls -lgd`. In addition, check the files in the spooling directory with `ls -lg`. The permissions, ownership, and group should match those shown below:

```
# ls -lg /usr/lib/lpd /usr/ucb/lpr
-rws--s--x 1 root daemon 53248 Oct 14 09:19 /usr/lib/lpd
-rws--s--x 1 root daemon 30720 Oct 14 09:19 /usr/ucb/lpr
# ls -lgd /var/spool/myprinter
drwxrwx--- 2 daemon daemon 512 Nov 09 11:00 /var/spool/printers/myprinter
# ls -lg /var/spool/lpd
-rw-r--r-- 1 root daemon 22 Mar 1 18:25 lock
-rw-rw-r-- 1 root daemon 29 Mar 1 17:28 status
# ls -lg /dev/ttya
crw-rw---- 1 daemon daemon 12, 0 Oct 21 11:57 /dev/ttya
```

- Make sure that `init(8)` does not create a login process on the port you are using for your printer. To do this, edit the `/etc/ttytab` file, making sure that the "status" field is "off" for the tty port that your printer is attached to. The example `printcap` attaches a serial printer to the first CPU serial port, `ttya`, `/etc/ttytab` should have an entry like:

```
ttya    "/usr/etc/getty std.9600" 925 off
```

If it was necessary to edit the `/etc/ttytab` file, you must notify `init` to make the system aware of the changes made. That is done by the following `kill` command while you are superuser:

```
# kill -HUP 1
```

Now you can send something to the printer with the `lpr` command. If the printer does not work now, but does if you send it output with the `cat` command to `/dev/tty*`, make sure the permissions and characteristics of all the `lpr`-related files are correct. They must be as described here. Make sure that your output filter (if you have one) is executable and that it is located where `/etc/printcap` says it is. Finally, check all the fields and the syntax in the `/etc/printcap` file.

12.5. Output Filters

The `printcap` examples above show the use of output filters. This section gives more details on their uses and specifications.

Filters are used to handle device dependencies and to perform accounting functions. The output filter `of` is used to filter data to the printer device when accounting is not used or when all data must be passed through a filter. It is not intended to perform accounting since it is started only once, all files are filtered through it, and no provision is made for passing owners' login names, identifying the beginning and ending of jobs, etc. The other filters, such as `if`, (if specified) are started for each job printed and perform accounting if there is an `af` entry.

They are used for producing the following types of jobs:

if plain text jobs

rf FORTRAN text files (i.e., jobs containing lines of text beginning with FORTRAN carriage control characters.)

- tf** troff jobs (i.e., jobs whose files contain Wang C/A/T codes, as produced by troff)
- nf** typesetter-independent troff jobs (i.e., jobs whose files contain ditroff output, as produced by the new typesetter-independent troff)
- df** TeX jobs (i.e., jobs whose files contain DVI codes, as produced by TeX)
- cf** CIF jobs (i.e., jobs containing CIF data, as produced by cifplot)
- gf** plot jobs (i.e., jobs containing plot codes, as produced by plot(1G))
- vf** raster output file jobs

If a job is to be printed, but there is no filter specified for that type of job, plain text jobs are run through the **of** filter, if any; other jobs are rejected and an error is logged.

The **af** entry designates the file where **if** puts its accounting informations. If entries for both **of** and one of the other filters are specified, the output filter is used only to print the banner page; it is then stopped to allow other filters access to the printer. An example of a printer that requires output filters is the Benson-Varian.

```
va|varian|Benson-Varian:\
:lp=/dev/va0:sd=/var/spool/vad:of=/usr/lib/vpf:\
:tf=/usr/lib/rvcat:mx#2000:pl#66:tr=\f:
```

The **tf** filter (invoked with `lpr -t`) takes a troff output file and converts it to Versatec output. It is used by `vtroff(1)`. Note that the page length is set to 66 lines by the **pl** entry for 8.5" by 11" fan-fold paper. To enable accounting, the **varian** entry would be augmented with an **af** file as shown below.

```
va|varian|Benson-Varian:\
:lp=/dev/va0:sd=/var/spool/vad:of=/usr/lib/vpf:\
:if=/usr/lib/vpf:tf=/usr/lib/rvcat:af=/var/adm/vaacct:\
:mx#2000:pl#66:tr=\f:
```

Output Filter Specifications

Sun software provides several filters that are listed under **CAPABILITIES** in `printcap(5)`. For many devices or accounting methods, it is probably necessary to create a new filter.

Filters are spawned by `lpd` with their standard input the data to be printed, and standard output the printer. The standard error is attached to the **lf** file for logging errors. A filter must return a zero exit code if there were no errors, 1 if the job should be reprinted, and 2 if the job should be thrown away. When `lprm` sends a kill signal to the `lpd` process controlling printing, it sends a **SIGINT** signal to all filters and descendents of filters. This signal can be trapped by filters that need to perform cleanup operations such as deleting temporary files.

Arguments passed to a filter depend on its type. The **of** filter is called with the following arguments.

```
ofilter -wwidth -llength
```

The *width* and *length* values come from the *pw* and *pl* entries in the *printcap* database. The *if* filter is passed the following parameters.

```
filter [-c] -wwidth -llength -lindent -n login -h host accounting_file
```

The *-c* flag is optional, and only supplied when control characters are to be passed uninterpreted to the printer (when the *-l* option of *lpr* is used to print the file). The *-w* and *-l* parameters are the same as for the *of* filter. The *-n* and *-h* parameters specify the login name and host name of the job owner. The last argument is the name of the accounting file from *printcap*.

All other filters are called with the following arguments:

```
filter -xwidth -ylength -n login -h host accounting_file
```

The *-x* and *-y* options specify the horizontal and vertical page size in pixels (from the *px* and *py* entries in the *printcap* file). The rest of the arguments are the same as for the *if* filter.

12.6. Access Control

The printer system maintains protected spooling areas so that users cannot circumvent printer accounting or remove files other than their own. The spooling areas are made up of one directory per entry in the local */etc/printcap*, under the directory */var/spool/area*. Here *area* can be any term specified in the *sd* (*spool directory*) entry in the */etc/printcap* database. The strategy used to maintain protected spooling areas is as follows:

- The spooling area is writable only by a *daemon* user and *daemon* group.
- The *lpr* program is run by *root*, which permits reading any file required. The group ID is used in setting up proper ownership of files in the spooling area for *lprm*.
- Control files in a spooling area have *daemon* ownership and group ownership *daemon*. Their mode is 0660. This ensures control files are not modified by a user and that no user can remove files except through *lprm*.
- The spooling programs *lpd*, *lpq*, and *lprm* are run with *root* permissions and group *daemon* permissions to access spool files and printers.
- The printer server, *lpd*, uses the same verification procedures as *rshd* (8C) in authenticating remote clients. The host on which a client resides must be present in the file */etc/hosts.equiv*.

12.7. Line Printer Administration

The `lpc` program lets you control the operations of those printers which are described in the `/etc/printcap` database. `lpc` can be used to start or stop a printer daemon, enable or disable the print queue for a printer, rearrange the order of jobs in a printer queue, or display the status of a printer, the printer daemon, and its spooling queue.

The basic syntax of `lpc` is as follows:

```
lpc [command [parameter...]]
```

With no arguments, `lpc` runs interactively, prompting with `lpc>`. If arguments are supplied, the first is interpreted as a command, and the following arguments, if any, are parameters to that command. For example:

```
example<25>: /usr/etc/lpc restart schroedinger
schroedinger:
    no daemon to abort
schroedinger:
    daemon started
```

This shows an attempt to start a spooling daemon for the printer `schroedinger` by performing the `restart` command.

Entering Commands and Parameters

`lpc` recognizes the following commands:

lpc Commands				
help	abort	clean	disable	down
enable	exit	quit	restart	start
status	stop	topq	up	?

You can give most commands to `lpc` some parameter. One such parameter is the printer's name for the `restart` command. When you specify a printer, any of the acceptable aliases of that printer are permissible (as defined by the `/etc/printcap` database). If desired, you can designate all available printers to be the subject of the command by using the word `all` in place of any specific printer. When no parameter is given to a `lpc` command, `all` is assumed as the default parameter in most cases.

Abbreviations

When entering a command to `lpc`, in either interactive or command line mode, you can abbreviate the command to an unambiguous abbreviation consisting of the first few letters of the command. In this fashion, the command `clean` may be shortened to `c` while `status` may only be abbreviated as short as `stat` in order to distinguish the command from `start`. If the abbreviated command that you have entered is ambiguous, `lpc` will respond with the message: `?Ambiguous command.`

Obtaining Help

`lpc` provides online help with the `help` command, which may also be entered as `?`. The `help` command gives a short description of each command in the argument list, or, if no arguments are given, a list of the recognized commands. Use the `help` command whenever you are uncertain as to the function of any `lpc` command.

Restarting a Printer

The most common need for `lpc` is when a printer has had its daemon (`lpd(8)`) die for any reason. When this occurs, all submitted jobs will pile up in the queue and not be printed. To start printer going again, you need to issue the `restart` command. When this is issued, `lpc` attempts to abort any existing daemon to avoid having two daemons running at the same time, and then starts a new daemon.

Determining When a Printer Daemon Has Died

Several methods can be used to find out if a printer has lost its daemon. From within `lpc` there is the command `status`, which shows the status of daemons and queues on the local machine. If no argument is given, then the command reports on status of all known printers. Outside of `lpc`, you can determine if a printer is running by first logging on to the machine which controls the printer and then using the `lpq`. The output from `lpq` will show you the pending print jobs, as well as notify you if any error conditions with the printer or the printer daemon. Another method to determine if a printer daemon is running is to log on to the machine which controls the printer and using the `ps` and `grep` commands as in the following example:

```
example<27>: ps -ax | grep lpd | grep -v grep
```

If no `/usr/lib/lpd` process is listed, then it is necessary to restart the printer daemon. If a `/usr/lib/lpd` process is listed, the a printer daemon exists and a `restart` is not necessary.

Exiting from `lpc`

You can use either the command `quit` or the `exit` command to finish using the `lpc` utility. The end of file character (`^D`) is also recognized as a command to leave `lpc`.

Other Printer Administration Commands

The other commands that `lpc` recognizes are usable only by the superuser. If you are not logged in as root and you attempt to perform any of these commands, `lpc` responds with the warning

```
?Privileged command
```

without doing anything else.

Enabling and Disabling printer queues

NOTE: If you disable a printer using `lpc`, this will also effect the System V style Printer Spooling System, if that system is running.

The first such privileged commands are the `disable` and `enable` commands. These commands are used to control the queues that store the jobs to be printed. When a `disable` command is issued, the printer queue is turned off— no jobs from that point on will be queued up. All jobs that are currently in the queue are unaffected. If anyone attempts to submit new jobs to the printer using `lpr`, the following warning is returned:

```
lpr: Printer queue is disabled
```

The `enable` command simply undoes the effects of the `disable` command.

Stopping and Starting Printer Daemons

Often it is necessary to stop a printer daemon, when you want to allow the current job to finish. This is accomplished with the `stop` command. When a `stop` command is given, `lpc` waits for the current printing job (if any) to complete. When this happens, the `lpd` daemon for that printer is killed.

To undo the effects of the `stop` command, the `start` command may be used. This command enables printing by starting a daemon for the listed printers. Note that the `stop` command does not disable queuing of jobs. Users may submit new jobs to a stopped printer, where the print jobs will wait in the queue until a new daemon is started.

Notifying users of a Disabled Printer

The `down` command works like the `disable` command, except that an optional message may be given as a parameter. This message will then be entered into the printer status file. This message will be shown when `lpq` or the `lpc` command `status` is used to determine a printers status.

The format for the `down` command is as follows:

```
lpc> down heisenberg Down for maintenance.
lpc>
```

The above command will disable queuing on the printer `heisenberg`, and will report “Down for maintenance.” for the status of that printer.

The `up` command undoes the effects of `down` by acting first to enable everything and then , if necessary, starting a new daemon.

Aborting a Printer Daemon

To ungracefully stop a printer in its tracks, the `abort` command is used. When this command is issued, the current printer daemon `lpd` is killed, and further printing is disabled.

Adjusting the Printer Queue

The contents of the printer queue can be adjusted using the `topq` and the `clean` commands. The `topq` command moves print jobs to the top of the queue (and therefore the jobs to be printed first). The jobs moved by `topq` are given by either listing a specific print job number, or by entering a user name. In the latter case, all print jobs submitted previously by that user are moved to the top of the queue.

The `clean` command deletes all files from the printer queue that begin with the letters `cf`, `tf`, or `df`. This command essentially deletes all pending print jobs from the queue.

12.8. Troubleshooting

There are several messages that may be generated by the line printer system. This section categorizes the most common and explains the cause for their generation. Where the message implies a failure, directions are given to remedy the problem.

In the examples below, the name *printer* is the name of the printer from the `printcap` database.

lpr

```
lpr: printer: unknown printer
```

The *printer* was not found in the `printcap` database. Usually this is a typing mistake; however, it may indicate a missing or incorrect entry in the `/etc/printcap` file.

```
lpr: printer: jobs queued, but cannot start daemon.
```

The connection to `lpd` on the local machine failed. This usually means the printer server started at boot time has died or is hung. Use `ls /dev/printer` to be sure that file still exists (if it does not exist, there is no `lpd` process running). Usually it is enough to get a superuser to type the following to restart `lpd`.

```
% /usr/lib/lpd
```

You can also check the state of the master printer daemon with the following.

```
% ps 1`cat /var/spool/lpd.lock`
```

Another possibility is that the `lpr` program is not `setuid` to `root`, `setgid` to group `daemon`. This can be checked with

```
% ls -lg /usr/ucb/lpr
-rws--s--x 1 root daemon 24576 Apr 9 1988 /usr/ucb/lpr
%
```

If you see:

```
lpr: printer: printer queue is disabled
```

this means the queue was turned off with

```
% lpc disable printer
```

to prevent `lpr` from putting files in the queue. You normally do this when a printer is

going to be down for a long time. You can turn the printer back on by using `lpc enable`.

lpq

waiting for *printer* to become ready (offline ?)

The printer device could not be opened by the daemon. This can happen for several reasons, the most common is that the printer is turned offline. This message can also be generated if the printer is out of paper, the paper is jammed, and so on. The actual reason is dependent on the meaning of error codes returned by system device driver. Not all printers supply enough information to distinguish when a printer is offline or having trouble (for example, a printer connected through a serial line). Another possible cause of this message is some other process, such as an output filter, has already opened the device, and that process has died or hung. Your only recourse here is to kill off the offending program(s), if it exists, and restart the printer with `lpc`.

printer is ready and printing

The `lpq` program checks to see if a daemon process exists for *printer* and prints the file *status* located in the spooling directory. If the daemon is hung, become superuser and use `lpc` to abort the current daemon and start a new one.

waiting for *host* to come up

This implies there is a daemon trying to connect to the remote machine named *host* to send the files in the local queue. If the remote machine is up, `lpd` on the remote machine is probably dead or hung and should be restarted as mentioned for `lpr`.

sending to *host*

The files should be in the process of being transferred to the remote *host*. If not, the local daemon should be aborted and started with `lpc`.

Warning: *printer* is down

The printer has been marked as being unavailable with `lpc`.

Warning: no daemon present

The `lpd` process overseeing the spooling queue, as specified in the lock file in that directory, does not exist. This normally occurs only when the daemon has unexpectedly died. The error log file for the printer and the `syslogd` logs should be checked for a diagnostic from the deceased process. To restart an `lpd`, use

```
% lpc restart printer
```

```
no space on remote; waiting for queue to drain
```

This implies that there is insufficient disk space on the remote machine. If the file is large enough, there will never be enough space on the remote (even after the queue on the remote is empty). The solution here is to move the spooling queue or make more free space on the remote machines spooling area.

lprm

```
lprm: printer: cannot restart printer daemon
```

This case is the same as when `lpr` prints that the daemon cannot be started.

lpd

The `lpd` program can log many different messages using `syslogd`. Most of these messages are about files that can not be opened and usually imply that the `printcap` file or the protection modes of the files are incorrect. Files may also be inaccessible if people manually manipulate the line printer system (that is, they bypass the `lpr` program).

In addition to messages generated by `lpd`, any of the filters that `lpd` creates may log messages using `syslogd` to the error log file (the file specified in the `lf` entry in `printcap`).

lpc

```
couldn't start printer
```

This case is the same as when `lpr` reports that the daemon cannot be started.

```
cannot examine spool directory
```

Error messages beginning with "cannot ..." are usually caused by incorrect ownership or protection mode of the lock file, spooling directory, or the `lpc` program.

Part Three: Network and Communications Administration

This part contains important procedures and theoretical information for administering local and wide area networks on the Sun workstation. The information in this part is applicable to any Sun workstation that is on a network.

Who Should Read This Part

If your machine is on a network, you should familiarize yourself with the procedures in a chapter that apply specifically to your system, be it a server or client, including a networked standalone. (Chapter 3 defines these terms from a networking perspective.) If you have a standalone or time-sharing system that is not attached to a network, Part Three does not apply to you.

What Is in This Part

Part Three discusses the following:

- What the TCP/IP network protocol is, steps you need to perform to get a network up and running, and suggestions for troubleshooting network-related problems.
- How to set up and administer servers and clients to use the Sun Network File System.
- How to set up and use the NFS automounter.
- How to set up and administer Domain Name Service.
- How to set up and administer a domain running the Remote File Sharing service.
- How to administer the Sun yp service.
- How to administer electronic mail on your system.
- How to set up and maintain uucp services.
- How to set up C2 security.

The SunOS Network Environment

This chapter explains how to set up and administer a network based on the TCP/IP Internet protocol family. The administrative steps needed are simple and few — once you have understood the concepts underlying them. Therefore, the chapter concentrates on the concepts that will allow you to install the most appropriate network for your organization. Its organization is as follows:

- "Introducing the Internet Protocol Family," which introduces basic concepts relating to TCP/IP and describes the tasks which an administrator must perform when dealing with a TCP/IP network.
- "Setting up Network Software," in which various aspects of the network software are discussed. Also covered are the procedures you follow to obtain a network number, register a domain, and boot TCP/IP on the network hosts.
- "Maintaining TCP/IP Files," which deals with the various administrative files that are needed to run TCP/IP properly.
- "Security in a TCP/IP Environment," which explains the effects of the `hosts.equiv`, and `.rhosts` files, as well as the implications that these files have on security issues. Furthermore, other aspects of security in the TCP/IP environment are covered.
- "Expanding your Network," in which the subject of what hardware is needed to expand your network are discussed. In addition, the procedures for creating an internetwork, setting up a router, and setting up subnets are shown.
- "Diagnosing Network Problems," in which various commands are listed which can be of assistance when problems relating to the network are encountered. These commands are explained and the methods in which they can be used are shown.

13.1. Introducing the Internet Protocol Family

A network is a configuration of machines that exchange information among themselves. In order for the network to function properly, the information originating at a sender must be transmitted along a communication line and delivered to the intended recipient in an intelligible form. Because different types of networking software and hardware need to interact to perform this function, network designers developed the concept of the communications protocol family (or suite). A *Network protocol* is a set of formal rules explaining how

software and hardware should interact within a network in order to transmit information. The Internet Protocol family is one such group of network protocols. It is centered around the Internet Protocol (IP). The other members of the Internet protocol family are Transmission Control Protocol (TCP), User Datagram Protocol (UDP), Address Resolution Protocol (ARP), Reverse Address Resolution Protocol (RARP), and Internet Control Message Protocol (ICMP).

The entire family is popularly referred to as TCP/IP, reflecting the names of the two main protocols. This is also the terminology followed by this document.

The Internet Protocol Suite

TCP/IP provides service to many different types of host machines connected to heterogeneous networks. These networks may be wide area networks, such as X.25-based networks, but also they also can be local area networks, such as you might install in a single building.

TCP/IP was originally developed by the United States Department of Defense. Many popular texts use the term "Internet" to describe both the protocol family and a particular wide area network. This text uses the term *TCP/IP* for the Internet protocol suite and *Internet* when referring to the network itself.

If you plan to run TCP/IP on your local network, you must obtain an IP network number and register your domain name with the Network Information Center (NIC) at SRI International (also referred to as SRI-NIC). A later section explains how to do this.

TCP/IP Protocol Structure

The TCP/IP protocol structure can be conceptualized as being formed of a series of layers as shown in Table 13-1.

Table 13-1 TCP/IP Protocol Layers

<i>Layer</i>	<i>Network Services</i>
Application	Domain Name Service, Telnet, others
Transport	TCP, UDP, ICMP
Network	IP
Data Link	ARP, RARP, Controller, such as Ethernet
Physical	Cable or other device

The primary function of the TCP/IP protocols is data communication. Every protocol layer on the sending host has its peer protocol layer on the receiving host. Moreover, each layer is required by design to handle communications in a pre-determined fashion. Each protocol formats communicated data and appends or removes information from it. Then the protocol passes the data to a lower layer on the sending host or a higher layer on the receiving host, as illustrated in the following figure:

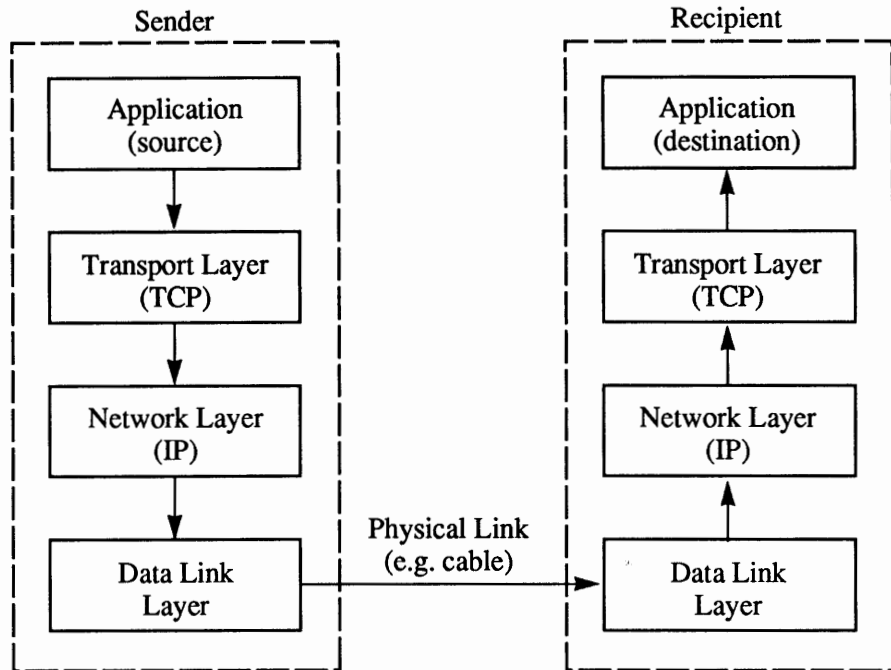


Figure 13-1 *Sender/Receiver Interaction*

The next subsections briefly explain how each protocol layer handles messages, from the lower to the higher layer. For more detailed information, refer to the man page for the appropriate protocol.

Physical Layer

The Physical layer is the hardware level of the protocol model, thus it is concerned with electronic signals. Physical layer protocols transmit data in the form of *packets*. A packet contains a source address, the transmission itself, and a destination address. The Physical layer sends and receives packets; packets also exist at the Data Link layer.

TCP/IP supports a number of Physical layer protocols, including Ethernet, IEEE 802.5 FDDI, Token Ring, and others.

Data Link Layer

The Data Link layer is concerned with addressing at the physical, machine level. Therefore, protocols at this layer are involved with communications controllers, their chips, and their buffers. Ethernet is an example of standards supported at this layer by TCP/IP.

Two additional TCP/IP protocols, ARP and RARP, can be thought of as existing between the Network and Data Link layers. ARP is the Ethernet address resolution protocol. It maps known IP addresses (32 bits long) to Ethernet addresses (48 bits long).

RARP (or Reverse ARP) is the IP address resolution protocol. It maps known Ethernet addresses (48 bits) to IP addresses (32 bits), the reverse of ARP.

ARP and RARP are not used when this kind of mapping is not needed, that is, when Ethernet is not in use.

Network Layer

IP (Internet Protocol) is the protocol present at the Network layer. It provides machine to machine communication. Most significantly, IP performs transmission routing by determining the path a transmission must take, based on the receiving machine's IP address. (The later section *IP Addressing* discusses IP addresses in detail.) IP also provides transmission formatting services: it assembles data for transmission into an *internet datagram*. If the datagram is outgoing (received from the higher layer protocols), IP attaches an *IP header* to it. This header contains a number of parameters, most significantly the IP addresses of the sending and receiving hosts. Other parameters include datagram length and identifying information, in case the datagram exceeds the allowable byte size for network packets and must be fragmented.

For incoming datagrams (received from the lower protocol layers), IP uses information in the header to identify the transmission and determine if it is a fragment of a datagram. If the transmission was fragmented, IP reassembles the fragments into the original datagram, which it passes on to the higher protocol layers.

Refer to the `ip(4p)` man page for more information.

Transport Layer

The TCP/IP transport layer protocols enable communications between processes running on separate machines. Protocols at this level are TCP, UDP, and ICMP.

TCP (Transmission Control Protocol) enables applications to talk to each other as though they had a physical circuit. TCP sends data in the form of *virtual circuits*. Virtual circuits appear to be transmitted between hosts in a character-by-character fashion, similar to the way a keyboard sends a byte of data representing a character to the CPU each time you press a key. A transmission consists of a starting point, which opens the connection, the entire transmission in byte order, and an ending point, which closes the connection.

TCP attaches a header onto the transmitted data. This header contains a large number of parameters, which help processes on the sending machine to connect to peer processes on the receiving machine. TCP uses *port numbers* as its addressing method, which enable differentiation between participants in a transmission.

TCP is a connection-oriented, reliable protocol: any data written to a TCP connection will be received by its peer, or an error indication will be given.

UDP (User Datagram Protocol) is the alternative protocol available at the Transport layer. UDP is a connectionless, unreliable datagram protocol. These datagrams are groups of information transmitted as a unit to and from the upper layer protocols on sending and receiving hosts. UDP datagrams use port numbers to specify sending and receiving processes. However, no attempt is made to recover from failure or loss; packets may be lost with no error indication given.

Whether TCP or UDP is used depends on the network application invoked by the user. For example, if the user invokes `telnet`, that application passes the user's request to TCP. If the user's request involves the domain name server, that

application passes the request to UDP.

Refer to the `tcp(4p)` and `udp(4p)` man pages for more information about these protocols.

The third protocol at the Transport layer is ICMP (Internet Control Message Protocol). It handles errors and is responsible for printing error messages. Refer to the `icmp(4p)` man page for more information about it.

Application Layer

A variety of TCP/IP protocols exist at the Application layer. Here is a description of some of the most widely used:

□ telnet

The Telnet protocol enables terminals and terminal oriented processes to communicate on a network running TCP/IP. It is implemented as the program `telnet` (local machine) and the daemon `in.telnetd` (remote machine). Telnet provides a user interface through which two hosts can open communications with each other, then send information on a character-by-character or line-by-line basis. The application includes a series of commands, which are fully documented in the `telnet(1C)` man page.

□ ftp

The File Transfer protocol (FTP) transfers files to and from a remote network. The protocol includes the `ftp` command (local machine) and `in.ftpd` daemon (remote machine). `ftp` lets you specify on the command line the host with whom you want to initiate file transfer, and options for transferring the file. The `in.ftpd` daemon on the remote host handles the requests from your `ftp` command. If the `in.ftpd` daemon is not present on the remote host, `ftp` invokes a command interpreter that provides a full set of options for file transfer.

The `ftp(1c)` man page describes all options to `ftp`, and commands you invoke through the command interpreter. The `ftpd(8c)` man page describes the services provided by this daemon.

□ tftp

The Trivial File Transfer Protocol (TFTP) enables users to transfer files to and from a remote machine. Like `ftp`, `tftp` is implemented as a program in the local machine, and as a daemon (`in.tftpd`) in the remote machine. `tftp` invokes a command interpreter for transferring files; however, `tftp` does not actually maintain a connection between two machines between file transfers. The `tftp(1c)` man page describes the commands you can give to the `tftp` command interpreter.

□ Domain Name Service (DNS)

This protocol provides domain name-to address-mapping of forwarding hosts and mail recipients on a network. The man page `named(8c)` describes the service. The chapter *Administering Domain Name Service* describes how to run this service on your network.

Other protocols exist that are also implemented as a program on the local machine and a daemon on the remote one; examples of these are `rlogin` and `in.rlogind`, which permit a user to log in to a remote machine; `rsh` and `in.rshd`, which enable the user to spawn a shell on a remote machine, and `finger` and `in.fingerd`, which permit a user to obtain information about users in remote machines.

To avoid the need to have an excess of daemons running at all times, the daemon `inetd` is initiated at startup time and, after consulting the `/etc/inetd.conf` file, it takes upon itself to run the appropriate daemons as needed. For example, the daemon `in.rlogind` will be run by `inetd` whenever there is a request for a remote login from another machine, and only at that time and for the duration of the `rlogin`.

Administration tasks

The administration of a TCP/IP based network consists of:

- Obtaining an IP network number for a new network; In order to connect a network to the Internet it is necessary to obtain a network number from SRI-NIC. This can be done by filling out the form in appendix F.
- Registering the domain name for a new network; To register a domain with SRI-NIC, you must fill out the form in appendix F. All new domains connecting to the Internet must be registered.
- Establishing the network and (optionally) the sub-networking configurations; The administrative arrangement of the network should be established, and administrative subdivisions may be desirable.
- Maintaining the database files for each host and service; Each network has a set of files which list the hosts and services using that network. These files should be maintained and kept current with each other. The information for doing these tasks is in *Maintaining TCP/IP-Related Files*.
- Maintaining the security of the network; The security of the network can be compromised if you do not take proper steps, such as creating and maintaining proper `hosts.equiv` and `.rhosts` files. See the later section, *Security in a TCP/IP Environment*.
- Diagnosing and debugging network problems.

13.2. Setting Up Network Software

This section explains how to set up and maintain network software, including:

- IP addressing and network classification
- Obtaining an Internet network number
- Setting up an administrative domain
- Installing TCP/IP

IP Addressing

A TCP/IP network routes a packet according to the destination *IP address* attached to it by the IP protocol on the sending host. Sometimes packets travel only from one host to another on the same local network. Or, they travel from one local network to another network, going through a router. In the extreme, a packet may transverse many networks, crossing miles of routers, gateways, cabling, and telephone lines to reach the destination IP address.

Addressing and Network Classification

To properly run TCP/IP, your network must have an IP network number, and every host on it must have an IP address. A machine's IP address consists of the following information:

<i>net number</i>	<i>[subnet number]</i>	<i>host number</i>
^	^	^
NIC assigns	You assign	You assign

Here is how you determine the numbers to be used in the IP address:

- *net number*
You obtain this from the NIC, the Network Information Center, as described in the subsection *Obtaining an IP Network Number*. You must not arbitrarily assign network numbers to your network.
- *subnet number*
This is an optional number representing the subnet to which the host is attached. This is a number that you (as opposed to the NIC) assign to the subnet when setting it up. You only state the subnet number if the machine is on a subnet. Refer to *Setting Up Subnets* for further information about subnet numbering.
- *host number*
This is a number that uniquely identifies this machine on your network. You assign this number.

IP Address Representation

All this information is represented by a 32-bit wide IP address, divided into four 8-bit fields. Each field is called an *octet*; information in it is represented by a decimal number separated from the next octet by a period. The decimal number represents one byte of the IP address, which can have a value of 0 to 255. Here is a typical IP address.

Note: All IP numbers in this manual are **examples only**. Do not use them for your own purposes.

129.144.50.56

Network Classes

Networks numbers are divided into three classes: A, B, and C. Each network class is specified in the IP address by a numeric value within a certain range.

- The Class A Network Number

There are very few Class A network numbers, but each accommodates a large number of hosts. Typically, Class A network numbers belong to large

organizations or universities. The values assigned to the first octet of Class A network numbers fall within the range 0-127. Consider the IP address 75.4.10.4. The value 75 in the first octet indicates that the host is on a Class A network. The remaining octets 4.10.4 establish the host address. Thus you can see how there are few Class A networks—only 127. The highest host address in a Class A network can be 255.255.254, thus accommodating a possible 16,777,214 hosts.

□ **The Class B Network Numbers**

Class B network numbers provide a median distribution between networks and hosts. The NIC assigns Class B addresses to medium-sized companies or institutions with many hosts on their networks. The first octet of a class B network is in the range 128-191. The first IP address example shown in this section, 129.144.50.56, is for a machine on a Class B network. The first two octets, 129.144, address the network. The last two, 50.56, address the host.

□ **The Class C Network Numbers**

Class C network numbers provide for many networks with few hosts on each—maximum 254. The first octet of a class C network is in the range 192-223. A typical Class C address might be 192.5.2.5. Here the first three octets, 192.5.2, address the network, and the final octet, 5, addresses the host.

□ **Subnets**

Subnetting allows you to subdivide a single Class A, B or C network number into many network numbers (subnets). A later section describes how to set up subnets.

The following table illustrates how addresses are structured:

Figure 13-2 *Network Address Structure*

	Range	Network Address	Host Address
Class A	0-127	xxx	xxx.xxx.xxx
Class B	128-191	xxx.xxx	xxx.xxx
Class C	192-223	xxx.xxx.xxx	xxx

Obtaining an IP Network Number

Always get an IP network number from the NIC at SRI International for any network that will run TCP/IP. You should contact the NIC even if your network will not connect to the Internet. This organization handles all assignments of network numbers.

To obtain an IP number, fill out to appropriate form and sent it to the NIC. Appendix F at the end of this document contains a copy of a form with instructions on how to fill it out and where to send it.

When registering, you must tell the NIC the classification (A, B or C) you want for your network. Choose the smallest class that will accommodate network growth over the next few years. For a fairly large organization that routes together local area networks in several buildings, consider requesting a Class B

address. This is a good idea, particularly if you plan to subnet part of your local networks. If your network will not conceivably support more than 255 hosts, ask for a Class C address.

Creating IP Addresses for Your Hosts

Once you have received your network number from the NIC, you can create IP addresses for your hosts. You may want to do this on paper or in an ASCII file for reference, before supplying these addresses to the programs and files that must use them.

The first part of the host's IP address must consist of the network number that the NIC assigned you. If the host is on a subnet, refer to the section *Setting Up Subnets* for information about IP addressing for this special case.

You assign the host part of the IP address in the remaining octets after the network number. In the unlikely event that your hosts are on a Class A network, you have three octets for defining the host address. On a Class B network, you have two octets, on a Class C network, one.

You may assign values from 0-255 in each octet allowed for the host address, with the restriction that the "host" part cannot be all zeros or all ones. In other words, if, for example, you have a Class A network where the net address is 10, a host could have the address 10.30.0.107 or 10.1.1.255, but it could not have the address 10.0.0.0 or 10.255.255.255. Similarly, if your network number is 192.9.90, you can not assign addresses 192.9.90.0 or 192.9.90.255 to any of its hosts.

The host address part of the IP address must be unique for every host on your network. Suppose you want to add a host named *dancer* on your Class C network. If the Class C network has the network number 192.9.200, you might assign the IP address

```
192.9.200.1
```

to *dancer*.

Then you might want to install other hosts on the same network with IP addresses such as 192.9.200.2, 192.9.200.3, and so on.

Establishing a Domain

The next step in setting up your TCP/IP network is creating its administrative domain. This involves the following activities:

- Establishing a name for the domain
- Registering the domain with outside upper level networks
- Updating the network administration files

Selecting the Domain Name

The first steps in setting up a domain is determining its name and which machines are part of it. Domain names should correspond to administrative divisions, not necessarily to how networks are connected.

Domain names are hierarchical, with components separated by periods. The top level domain is the component on the furthest right of the name; lower level

domains follow from right to left. This hierarchical system is similar to full pathnames in UNIX, except that UNIX pathnames are read in the opposite direction — from left to right. (Refer to *Administering Domain Name Service* for more information on how domains are named.)

Your full domain name must be unique within the world. For example, you could use “podunk.EDU” as a base name for a domain consisting of all machines at Podunk University, provided that another university has not used this name first. Upper and lower case letters are not significant in domain names. The traditional UNIX convention is to use all lower case; many other systems use upper case.

A full domain name must begin with a valid top-level domain. Top level domains often define the type of administrative entity they apply to. Some valid top-level domains are:

.COM	Commercial Companies
.EDU	Educational Institutions
.GOV	Government Agencies

For example, Podunk University’s domain name would begin with “.EDU”.

Domains are not limited to any fixed number of levels. For example, the Amalgamated Widget Company might register the domain name “Widget.COM”. It might have a subdomain called “eng.Japan.Widget.COM” for the engineering group within their Japan subsidiary, and another called “mktg.Japan.Widget.COM” for marketing in Japan. However, machines at company headquarters might belong to the subdomain called “HQ.Widget.COM.”

Registering Your Domain

After choosing the name of your domain (“Widget.COM” in the example above), and before setting it up, you need to register it with a higher level domain. To do this, you should get in touch with the administrator of that domain. The top-level domains are administered by the NIC at SRI International.

Appendix G provides you with a copy of the Domain Registration Form.

Once your domain is registered, you can divide it into subdomains as necessary.

Setting a Host’s Domain Name

Each host in your new administrative name must know its domain name. This name is initially set through SunInstall at installation time, and recorded in the file `/etc/defaultdomain`. Subsequently, whenever you boot the machine, the `/etc/rc.local` file executes:

```
domainname `cat /etc/defaultdomain`
```

To change a machine’s domain name, or if it has not been set at installation time, you have to change the content of `/etc/defaultdomain` with the new name and reboot the machine. If you are doing this for a diskless machine while logged in on the server, the file is by default in the directory `/export/root/client/etc`. The content of the file is the name of the (sub)domain to which the host belongs.

Booting TCP/IP on Your Network Hosts

Each machine starts TCP/IP through the `/etc/rc.boot` and `/etc/rc.local` scripts, where `ifconfig` is run. Among its many functions, `ifconfig` defines the network address of each network interface present on a machine, or redefines it. In the `/etc/rc.boot` file the line

```
ifconfig interface hostname netmask + -trailers up
```

is executed for each interface present in the form of a file of the name `/etc/hostname.??#`, where `??#` represents the name of an interface (such as `le0` or `ie1`, for instance) and where the hostname is taken from the contents of that file. This, however, is not done on diskless machines, which boot through the net, already have everything set, and do not have `/etc/hostname.??#` files.

In `/etc/rc.local`, the netmask and the broadcast are reset for each network interface in the line

```
ifconfig -a netmask + broadcast + > /dev/null
```

See below for an explanation of netmask.

You can query `ifconfig` for a display of the current configuration of a particular network interface, or, if called with the `-a` option, of all network interfaces on the machine.

For a thorough discussion, please consult the `ifconfig(8)` man page.

13.3. Maintaining TCP/IP-Related Files

You use the SunOS *network databases* for administering your TCP/IP-based network. These databases and their contents are described below:

Database	Contents
hosts	Host names and IP addresses.
ethers	Host names and their Ethernet addresses.
networks	Network names and their numbers.
protocols	IP protocol names and their numbers
services	IP service names and their port numbers.
netmasks	Network names and their netmasks.

This section discusses all the databases listed above except `netmasks`, which is covered in the section *Expanding Your Network*.

The network databases can be accessed in at least two forms: files in the directory `/etc` or database distribution services on networks that run this kind of service (such as NIS or the Domain Name Service).

This section assumes that your network databases exist as files in the directory `/etc` and all the examples are geared towards this assumption; if this is not the case, you will have to gear your actual changes in the databases to the particular name service you run. The general discussion, however, applies in all cases.

The `hosts` Database

The `hosts` database is one of two databases containing the addresses of machines on your network. It is stored on each machine in the `/etc/hosts` file (unless you have a database distribution service running, as mentioned above).

The `hosts` database lists all machines in the local domain by IP address and name. Its format is

```
IP-number      host-name      nickname      #comment
```

Adding IP Addresses to the `hosts` Database

It is essential, for the proper functioning of the network, that all machines know the addresses of all other machines, and that there are no duplicated addresses.

When you initially set up your network you must specify the IP addresses of all the machines in the `hosts` database that each machine consults (including the address of the host itself). Subsequently, when you add a host to the network, you have to add its name to the `hosts` database; this means that if the `hosts` database is accessed as a file in the `/etc` directory, you have to modify the `/etc/hosts` file on each machine to reflect all changes. If, on the other hand, the `hosts` database is accessed through YP or DNS, you do not have to modify the `/etc/hosts` files on all machines. But each machine should have an entry for itself, for `localhost` and, if diskless, for its file server.

The following is a sample `/etc/host` file:

```
192.9.200.1    dancer
192.9.200.2    ballet    # This is a comment
192.9.200.3    raks
192.9.200.4    samba
127.0.0.1     localhost localhost
```

You can also add any nicknames for a host next to its official host name. As you might suspect, a *nickname* is another name by which the host can be referred to. You can also add a comment to each entry.

Make sure also that you never delete by mistake the line:

```
127.0.0.1    localhost
```

The “localhost” address above is always the same, and is used by programs to reach services on the same machine from which they are invoked.

For more information about `/etc/hosts`, refer to the `hosts(5)` man page. For more information on NIS, refer to *The Network Information Service*; for more information on DNS, refer to *Administering Domain Name Service*.

The ethers Database

The `ethers` database associates host's names with their Ethernet addresses. This database is needed exclusively when the interface used is Ethernet. Along with `hosts` it permits locating machines on the network. It is used by the RARP daemon to map Ethernet addresses to IP addresses. It is accessed through the `/etc/ethers` file or the `ethers` NIS maps.

`/etc/ethers` lists, by Ethernet address and name, all machines in the local domain. Like `/etc/hosts`, you need to create `/etc/ethers` when you set up the network, and maintain it in all hosts to reflect any changes, unless your network is running NIS, in which case you update the `ethers.byname` database on the master server.

Here is a sample `/etc/ethers` file such as you might find on your system.

```
#
# Entries in this file should be in alphabetical
# order by machine name.
#
8:0:20:1:40:14  ballet
8:0:20:1:2h:7   dancer      # This is a comment
8:0:20:1:40:15  raks
8:0:20:1:40:16  samba
```

The first column of the file contains the Ethernet address of each host. This address is displayed whenever you power up your machine and whenever you boot it up. You can probably obtain it also through the command

```
example% dmesg
```

Look for a line that says something like:

```
Ethernet address = 8:0:20:7:37:32
```

If you can't find this line in `dmesg`'s output, like in the file `/var/adm/messages`.

The second column in `/etc/ethers` contains the official host names of each machine. When adding entries to `/etc/ethers`, make sure that these host names correspond to the primary names in `/etc/hosts` (not the nicknames).

Refer to the `ethers(5)` man page for additional information.

The networks Database

The `networks` database contains the names of all TCP/IP-based networks in your internetwork. It is represented by the file `/etc/networks`.

The `networks` database is used to associate a network number to a name. Your database should contain all the information relevant to your domain.

Here is a sample `/etc/networks` file:

```

#
#
# Internet networks
#
arpanet      10          arpa
# local networks
loopback    127
eng         193.9.0        #engineering
acc         193.9.1        #accounting
prog        193.9.2        #programming

```

You need to update `/etc/networks` to reflect all networks that your machine can use. This is particularly important because the `netstat` program described later in the section on *Diagnosing Network Problems* uses the information in `/etc/networks`.

You need to update `/etc/networks` on the following occasions:

- You just installed a router and you want to tell your machine about the other network to which the router is attached (see the section *Expanding Your Network*.)
- You join a wide-area network which is not listed in the file.

To modify `/etc/networks`, represent each network you want to add with a single line entry in the file. Use the following syntax:

```

official_network_name network_number nickname(s) #comments

```

official network name is the name by which the network is known. *network number* is the network's IP network number. *nickname* is any other name by which the network is also known. You can also comment out a line by prepending a # mark.

The protocols Database

The protocols database contains the names of the TCP/IP protocols for use by various programs. Its contents are represented in the file `/etc/protocols`. You do not maintain this file, though you may want to display it for informational purposes. Below is an example `/etc/protocols` file:

```

#
# @(#)protocols 1.8 88/02/07 SMI
#
# Internet (IP) protocols
# This file is never consulted when yp is running
#
ip 0 IP # internet protocol, pseudo protocol number
icmp 1 ICMP # internet control message protocol
igmp 2 IGMP # internet group multicast protocol
ggp 3 GGP # gateway-gateway protocol
tcp 6 TCP # transmission control protocol
pup 12 PUP # PARC universal packet protocol
udp 17 UDP # user datagram protocol

```

Refer to the protocols(5) man page for more information.

The services Database

The services database contains an entry for each TCP/IP network service available on your system. It is represented by the /etc/services file. Various programs read the services database; you do not maintain it.

Here is an excerpt from a typical /etc/services:

```

#
# @(#)services 1.12 88/02/07 SMI
#
# Network services, Internet style
# This file is never consulted when yp is running
#
echo          7/udp
echo          7/tcp
discard      9/udp          sink null
discard      9/tcp          sink null
systat       11/tcp
daytime      13/udp
daytime      13/tcp
netstat      15/tcp
ftp-data     20/tcp
ftp          21/tcp
telnet       23/tcp
smtp         25/tcp          mail
time         37/tcp          timserver
time         37/udp          timserver
name         42/udp          nameserver
whois        43/tcp          nicname          # usually to sri-nic

```

Notice that various network services like ftp and telnet are listed in the first column. The second column contains the service's port number and the transport layer (either TCP or UDP) protocol that handles it. Refer to the services(5) man page for more information.

13.4. Security in a TCP/IP Environment

The SunOS operating system includes various user programs that enable you to perform operations on remote hosts. These commands, also known as *r-commands* (`rlogin`, `rsh` and `rcp`) let you log in, create a remote shell, and copy files to and from a remote machine.

Normally, remote users must supply their local user names and passwords in order to gain access to your machine via the r-commands. However, you can set up two files to grant remote users access to your machine without their having to supply passwords. These files are:

```
/etc/hosts.equiv
/home_directory/.rhosts
```

This section explains how to administer these files to allow or deny remote access to your host through the r-commands. Note the other commands such as `telnet`, `ftp` or `tftp` are not affected by these files. The text assumes you are familiar with the local files `/etc/passwd` and `/etc/group`, which are described in the man pages and in other chapters throughout this manual.

The `/etc/hosts.equiv` File

The `/etc/hosts.equiv` file establishes whether a given machine is to be treated as a trusted host or not, and whether a given user at that machine will be granted access to the local machine or not.

A typical `hosts.equiv` file has the following structure:

```
host1
host2 user_a
+@group1
-@group2
```

When a simple entry for a host is made in `hosts.equiv`, such as the entry above for `host1`, it means that the host is trusted, and so is any user at that machine.

If a user name is also mentioned, as in the second entry above, then the host is trusted only if the mentioned user is attempting access. Also in the above example, all machines in `group1` are considered trusted, while all machines in `group2` are not. Groups are defined in the `netgroup` database (see below and *The Network Information Service*). A single `+` on a line in a machine's `hosts.equiv` file means that all known hosts are trusted.

Refer to the `hosts.equiv(5)` man page for more information.

The `.rhosts` File

The `.rhosts` file is located in the user's home directory, and it is consulted, if needed, after `hosts.equiv`.

The format of `.rhosts` is the same as that of `/etc/hosts.equiv`, and its function is to refine the trusted hosts list in `hosts.equiv`.

If an entry in `hosts.equiv` denies a user access to the machine, but an entry in the user's `.rhosts` grants it, then the user has access.

The netgroup database

The `/etc/netgroup` file, or its equivalent NIS map, lists network groups of machines. The format of entries in it is:

```
netgroup  name    name    ...
netgroup  (hostname, username, domainname) ...
```

That is, a netgroup is defined as a list of names, where the names can be either another netgroup or a triplet that completely describes the group member. An example may make things clearer:

```
forestry  willow  cedar  pine
willow    (willow, john, interflora.com)
cedar     (cedar, rose, interflora.com)
pine      (pine, marge, interflora.com)
```

If the `/etc/hosts.equiv` file in host oak has an entry saying

```
+@forestry
```

then users will be able to access oak from the machines willow, cedar and pine.

The `netgroup(5)` entry in *SunOS Reference Manual* completely describes this file,

The assumption that a triple specified as `(, , domain)` limits/grants access to any machine or user within that domain is erroneous. An `r-command` (`rlogin`, `rsh`) does not pass any parameter indicating which domain it is coming from, and therefore no verification is possible. The machine assigns its own domain name to any other machine attempting to connect to it, then checks the netgroup database for verification.

For example, if the netgroup `forestry` was defined as `(, , interflora.com)`, and an entry was placed in machine oak's `/etc/hosts.equiv` saying `+@forestry`, all machines will be given trusted host capability, regardless of their effective domain, if oak belongs to that domain; if it belongs to another domain, all requests will be denied.

How the Administrative Files Affect Network Security

The following diagram illustrates how the `hosts.equiv` and `.rhosts` files affect network security in relation to the `r-commands`. The diagram does not attempt to cover group entries in those files.

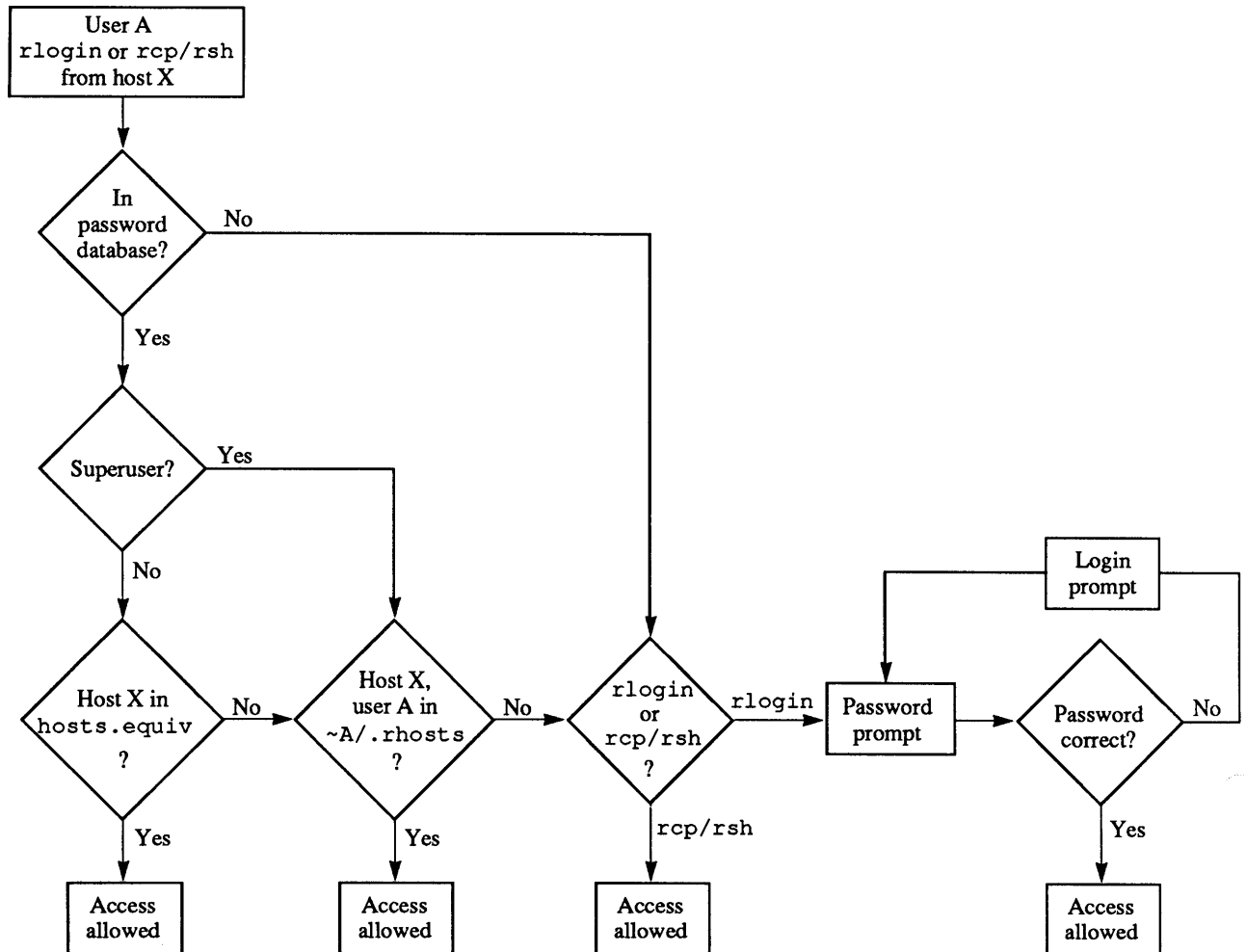


Figure 13-3 *rlogin/rcp/rsh Security*

Notice how `rlogin` is treated differently from `rsh` and `rcp`, and the widely divergent paths depending on whether there is an entry for the user in the password database or not.

The following points complement the above diagram:

- If a netgroup is denied access through a negative entry in `hosts.equiv`, but a user has a positive entry for that netgroup in his/her `.rhosts` file, the group is considered trusted if the user attempts access from one of the machines that belong to it.
- If a netgroup is denied access through a negative entry in `hosts.equiv`, but a user has an entry for a machine that belongs to that netgroup in his/her `.rhosts` file, the machine is considered trusted if the user attempts access from it.
- If the user does not have a home directory in the machine, then `/.rhosts` is consulted.
- If the user is the superuser, only `/.rhosts` is consulted, never `hosts.equiv`.

Other Security Issues

Some programs can be seen, particularly if your network is connected to other networks or to the Internet, as problematic from a point of view of strict security. For instance, the finger daemon `in.fingerd` will tell anyone who asks the username and full name of everyone who is logged into a machine. There is no authentication of the requester and no auditing of the requests. Similarly, the `in.rwhod` daemon (on whose information `ruptime` bases its reports) freely passes around information about who is logged in a machine.

There are no files or databases to restrict access to the information provided by these daemons. The solution, for those wishing a higher level of security, is not to run these programs (comment out the appropriate line in the file `inetd.conf` by prepending a `#` at the beginning of line.) Another reason not to run the `in.rwhod` daemon is that it can degrade performance considerably.

13.5. Expanding Your Network

In time, you may need to attach more hosts to your network than you have allowable addresses. Or you might want two networks in different buildings to share resources and to be administered as a single domain. This section first discusses some of the hardware aspects of connecting networks or subnetworks, and discusses two ways you can expand your existing network:

- Connecting two or more networks, to create an internetwork.
- Creating a subnet of your network.

Hardware on a TCP/IP-based Network

The TCP/IP protocol family provides intercommunication among host computers, terminal servers, and other equipment on one or more local area or wide area networks. A typical local network serves a limited area—within a building or between neighboring buildings. Hosts attached to the local network may function as file servers, mail servers, print servers, terminals, and workstations. A network medium such as Ethernet cabling, or IEEE token ring or token bus, connects this equipment.

Some organizations expand their networks by linking local networks together via a computer called an *IP router*. A network configuration consisting of several networks linked together by routers is often called an *internetwork*.

Be careful not to confuse the term internetwork with the Internet. Your company can set up an internetwork if it needs to expand communications services. Furthermore, you, or one of your co-workers, may have the responsibility of managing it. By contrast, the Internet is the name of a particular internetwork.

Refer to the hardware manuals that came with your computer and communications media for information about connecting them as a network. Also, refer to the chapter on *Administering Electronic Mail*, for information about configuring mail servers, and the chapter on *Maintaining Printers and Print Servers* for information about print servers. The chapters *The Sun Network File System Service* and *The Remote File Sharing Service* contain information about setting up NFS and RFS file servers, respectively.

Hardware Devices for Expanding the Local Network

When your local network no longer meets your needs, you may want to expand it into an internetwork. Below is a description of the hardware necessary to create an internetwork from two or more local networks.

- **Repeater:** This is a device used at the Physical Layer of the network; it connects two networks (or subnetworks) together and copies each bit of a packet from one to the other. (Packets are groups of message data, as explained above, in the section *TCP/IP Protocol Structure*.)

There are inherent limitations in the use of repeaters, given that they copy *all* data from one network to the other. Implementations like Ethernet, which require that data traverse the network within a specific amount of time, impose a maximum allowable length and number of repeaters.

- **Bridge:** This is a device used at the Data Link layer of the network protocol model. It selectively copies packets from one network (or subnetwork) to another.

Bridges differ from repeaters in several ways. Because copying done by bridges is selective, this reduces traffic on each section of the network. Since bridges copy whole packets, the geographical or timing constraints of the basic physical network can be extended.

Like repeaters, bridges copy raw packets, a scheme which works for all protocols above the Data Link layer. Since bridges copy whole packets, in theory, connections like this are “invisible” to the software on all the machines on the network.

- **Router** (sometimes called a gateway): This is a device that forwards packets of a protocol family, in this case TCP/IP, from one logical network to another. (Note that a router may also be an IP host.) A logical network makes sense only to a particular protocol. Usually there is a one-to-one mapping between physical network and logical network, but subnets (explained in a later section) and bridges are exceptions to this rule. An internetwork is a collection of logical networks (all using the same protocol) connected via routers.

The router may forward packets between different physical types of networks, for example, from an Ethernet to a ring network. It can also forward packets between two logical networks of the same type, for example, between two Ethernets on different floors of a building.

During forwarding, a router looks inside the packet to find the destination address, then consults its routing table, which is normally kept up to date by having routers communicate with each other via some routing protocol. Note that it is possible to have a multilingual router—a single device that forwards packets for several protocol types, for example, TCP/IP, ISO, or XNS.

- **Application Gateway** (sometimes called a relay or forwarder): This device and associated software enable networks using different protocols to communicate with each other. Because it translates protocols existing at all layers of the protocol model, you can use gateways to connect networks that differ on all layers from each other.

You can also use an application gateway to connect parts of the same physical network that are using different protocols.

Creating an Internetwork


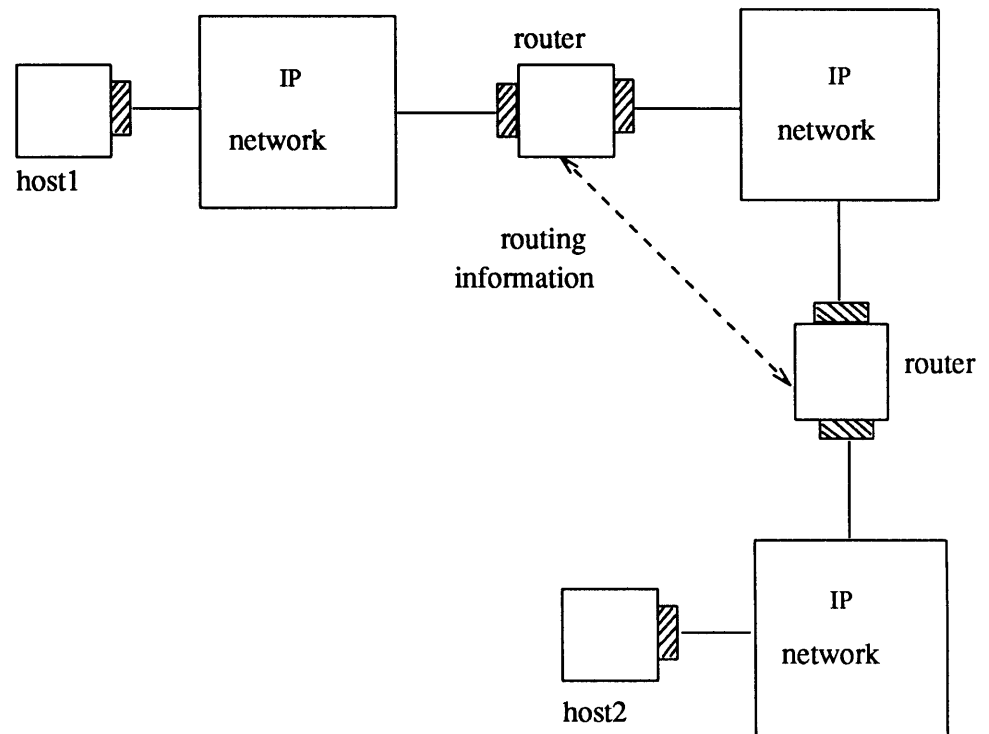
An IP network usually interconnects a number of hosts. Your machine is connected to an IP network via a hardware network interface. Individual IP networks are in turn interconnected via IP routers. IP routers forward IP packets from one IP network to another, and exchange routing information with each other to effectively deliver packets across a number of networks. Other types of routers may forward traffic for protocol families other than IP.

If all of the hosts at your site are connected to a single IP network that is *not* connected to any other IP networks, an IP router is unnecessary. If your site comprises many IP networks, or you wish to connect your IP network to other IP networks, you must engineer their interconnection with IP routers in order for all hosts to communicate.

Many types of machines may serve as IP routers. A number of vendors offer machines dedicated entirely to the function of IP routing. A machine running SunOS Release 4.1 will be automatically configured as a router if it has more than one real network interface. (All machines have at least one network interface, the *loopback* interface, which delivers packets within the local machine. To be configured as a router, a machine must have two additional hardware network interfaces.) In this configuration, the SunOS system will act both as a host (offering network services such as remote login) and a router. The `routed` (routing daemon) program implements a standard routing protocol in Release 4.1. If a machine has only one network interface, `routed` will passively monitor the routing traffic (if that network is a broadcast network). If a machine has two or more network interfaces, the `routed` program will actively participate in the exchange of routing information with other routers. You may choose to run other routing programs if your site employs a different routing protocol. The following figure illustrates the relationships between routers, hosts and networks:

Figure 13-4 *Routers and Networks*

LEGEND:

 hardware network interface


Configuring a Router

Configuring a router first involves setting up its hardware. Refer to the manuals that came with the machine for information on physical assembly. Once the router is connected to the networks it joins, you then must configure the router's software.

Before actually configuring this software, make the following preparations.

- Make sure you have acquired registered IP network numbers for each network the router is to connect.
- Assign the router a hostname and IP address for each network it is on. The Internet Protocol architecture requires each interface to have a unique IP address.

Assuming that you store the network databases in local files:

1. Log in as superuser on each host in the newly combined internetwork.
2. Add an entry in `/etc/networks` on each host for the new (other) network (see above, the section on *The networks Database*.)

3. Add an entry in each `/etc/hosts` for the router's host name and address on the second (or other) network.

For example, consider a router called `jeekyll` that connects two networks. Like all routers, `jeekyll` must have two hardware network interfaces. Before you edit it, the `/etc/hosts` file on machines in `jeekyll`'s network might resemble the following:

```
# Local Net 192.9.200 -- 10Mb/s Ethernet -- Engineering
#
192.9.200.1      jeekyll
192.9.200.2      usher
192.9.200.3      lenore
192.9.200.5      raven
# Local Net 192.9.201 -- 10Mb/s Ethernet -- Marketing
#
192.9.201.11     quasimoto
192.9.201.12     godzilla
192.9.201.13     rodan
```

To configure `jeekyll` as a router for the Marketing network 192.9.201, add an entry for its address on network 192.9.201:

```
# Local Net 192.9.200 -- 10Mb/s Ethernet -- Engineering
#
192.9.200.1      jeekyll
192.9.200.2      usher
192.9.200.3      lenore
192.9.200.5      raven
# Local Net 192.9.201 -- 10Mb/s Ethernet -- Marketing
#
192.9.201.11     quasimoto
192.9.201.12     godzilla
192.9.201.13     rodan
192.9.201.4      jeekyll-hyde
```

Notice that on the second network, `jeekyll` has another name, "`jeekyll-hyde`." The router still has the primary name `jeekyll`. But for the network software to work properly, the router must have a unique name and address for each interface. For ease of administration, use similar host names for each network. Users on both networks can address the machine by the primary name `jeekyll`, while you can tell the difference between the two.

4. Log in to the router as superuser and edit `/etc/hosts`; add the entry above, as you did for the other hosts on the network. The resulting `/etc/hosts` should now contain both of the router's IP addresses and host names, as shown below:

```
192.9.200.1 jeekyll
192.9.201.4 jeekyll-hyde
```

5. Create a file in `/etc` with the name `hostname.xx#`, where `xx` stands for the interface name, such as `ie` or `le`, and `#` stands for its number, and write in the file the name of the host. For instance, if “jekyll-hyde” is the host and the second interface is `ie1`, you could do:

```
# cat > /etc/hostname.ie1
jekyll-hyde
^D
#
```

Setting Up Subnets

Subnets are logical subdivisions of a single TCP/IP network. For administrative or technical reasons, many organizations chose to divide one network into several subnets. Subnets allow you more flexibility when assigning network addresses. The Internet Protocol allows 127 Class A networks with 24 bit host fields; 16,383 Class B networks with 16 bit host fields; and over two million Class C networks with eight bit host fields.

Routing can get very complicated as the number of networks grows. For example, a small organization might give each local network a Class C number. As the organization grows, administering network numbers may get out of hand. A better idea is to allocate a few Class B network numbers for each major division in an organization, for instance: one for Engineering, one for Operations, and so on. Then, divide each Class B network into physical networks using subnets. In this way, you can isolate hosts from changes you might make to the network in remote parts of the organization.

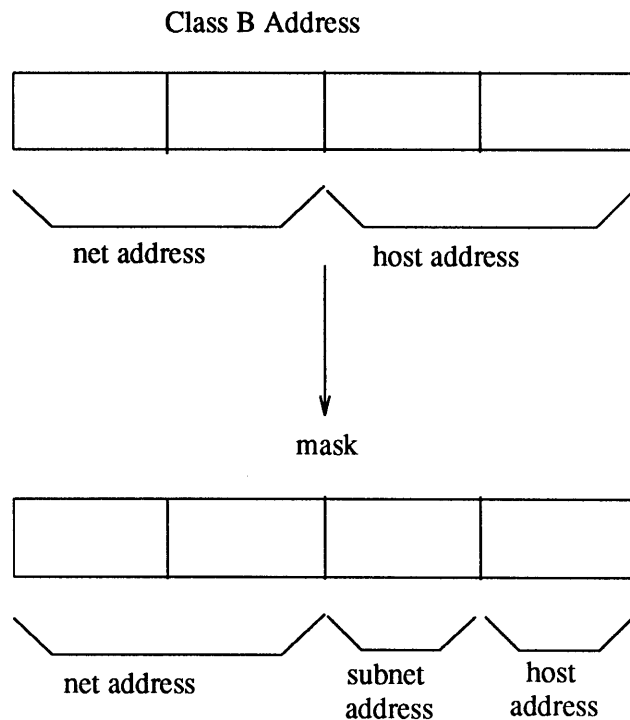
Network Masks

When setting up your network, you should select a network-wide *network mask*. It determines which bits in the IP address will represent the subnet number. The remaining bits will represent the host within the subnet. For example, you could configure an organization’s internetwork as a Class B network. Then you could assign each local subnetwork a subnet number within that network. The 16 bits could be allocated as eight for subnet and eight for host, or nine for subnet and seven for host, and so on. Your decision would be transparent to everyone outside that organization.

Routing within a subnetted network is based on *network#,subnet#* pairs; outside the network it is based on *network#* alone. The subnet mask is used to dissect an IP address into *network#,subnet#* and *host#* parts. This reduces network traffic, and improves the overall performance of your internetwork.

The following figure illustrates the effect of the network mask:

Figure 13-5 Network Mask



You can express network masks as four octets of decimal numbers, as described earlier in this book or, preferably, as a single hexadecimal number. The default is a mask of 0xFF000000 (255.0.0.0) for Class A networks, 0xFFFF0000 (255.255.0.0) for Class B networks, and 0xFFFFF000 (255.255.255.0) for Class C networks. You have to explicitly specify network masks only when they are wider (that is, have more one-bits) than the default values. One common case is a Class C mask on a Class B network. This provides you with 256 possible subnets, each one of which can accommodate 254 possible hosts (remember, 0 and 255 are not acceptable host addresses). But you may know that none of your subnets will ever have more than, say, 128 hosts, while you may need more than 256 subnets. In that case, you could decide to use nine bits for the subnet number instead of eight, and seven for the host addresses. The appropriate mask for this would be 0xFFFFF80, or 255.255.255.128 (2 to the power of 7 is 128, and 128 subtracted from the possible 256 is 128.)

Given the above scheme, and a network address of, for instance, 131.60, the address for the first host of the first subnet would be 131.60.0.129.

Subnetting a Class A or Class C network is similar to the above, once you make the corresponding adjustments.

The netmasks database

The `netmasks` database contains the default netmask for your system. It is represented by the file `/etc/netmasks`. To set up the netmask, you need to edit this file, which is described in the `netmasks(5)` man page.

Routers (machines with interfaces on more than one subnet) that have also NIS clients need to have the `/etc/netmasks` file as well as the NIS map `netmasks.byaddr`; that is, the file should be on the NIS master, and on each router for use during booting.

Here is a sample `/etc/netmasks`.

```
#
# Network masks database
#
# only non-default subnet masks need to be defined here
#
# Network      netmask
128.32.0.0    255.255.255.0
```

Create an entry with the network number and network mask on a separate line for each network that is subnetted.

You can use `ifconfig` to manually override the network masks. In an NIS environment, it is preferable to set the netmask with `ifconfig`. For more information about `ifconfig`, refer to the `ifconfig(8c)` man page.

For example, consider Class B network 128.32 with an eight bit wide subnet field (and, therefore, an eight-bit wide host field). The `/etc/netmasks` entry for this network would be:

```
128.32.0.0 255.255.255.0
```

You can enter symbolic names for subnet addresses in the `/etc/hosts` file. You can then use these subnet names instead of numbers as parameters to commands.

Changing from a Non-subnetted to a Subnetted Network

Follow these steps to change from an internetwork that does not use subnets to one that is subnetted.

1. Decide on the new subnet topology, including considerations for subnet routers and locations of hosts on the subnets.
2. Assign all subnet and host addresses.
3. Edit `/etc/netmasks` as mentioned previously.
4. Edit `/etc/hosts` on all hosts to change host address.
5. Reboot all machines.

Examples of Subnets

The following examples show network installations where subnets are (and are not) in use:

```
128.32.0.0 Berkeley class B network (subnetted) netmask 255.255.255.0
36.0.0.0 Stanford class A network (subnetted) netmask 255.255.0.0
10.0.0.0 Arpanet class A network (non-subnetted) netmask 255.0.0.0
```

All of the University of California at Berkeley is assigned the network number 128.32.0.0, so that any external router only needs to know one route to reach Berkeley. Within the campus, a class C subnet mask is used to give each local network a subnet number, with 254 hosts on each of the 254 possible subnets. (Zero and all ones, that is 255, are reserved.) Stanford University uses a class A network number with a Class B network mask, for 254 subnets of 65534 hosts each.

13.6. Diagnosing Network Problems

TCP/IP-based networks have their share of problems, as to be expected on any computer network. This section contains commands that can help you determine what is the nature of the problem you are having.

You can use several commands to troubleshoot problems that may arise on your network. These commands are:

```
ping
ifconfig
netstat
route
```

This section describes these commands and gives suggestions for using them.

The ping Command

The ping command offers the simplest way to find out if a host on your network is up. Its basic syntax is:

```
/usr/etc/ping host [ timeout ]
```

where *host* is the hostname of the machine in question. The optional *timeout* argument indicates the time in seconds for ping to keep trying to reach the machine— 20 seconds by default. The ping(8c) man page describes additional syntaxes and options.

The ping program sends an ICMP datagram to the host you specify, asking for a response. (Recall that ICMP is the protocol responsible for error handling on a TCP/IP network.)

Suppose you typed:

```
% ping elvis
```

If host elvis is up, you receive the following message:

```
elvis is alive
```

indicating that elvis responded to the ICMP request. However, if host elvis is down or cannot receive the ICMP packets, you will receive the following response

from ping:

```
no answer from elvis
```

If you suspect that a machine may be losing packets even though it is up, you can use the `-s` option of `ping` to try and detect the problem. For example, suppose you type:

```
% ping -s elvis
```

`ping` then continually sends packets to host `elvis` until you press **CONTROL-C**. Your responses will resemble the following:

```
PING elvis: 56 data bytes
64 bytes from 129.144.50.21: icmp_seq=0. time=80. ms
64 bytes from 129.144.50.21: icmp_seq=1. time=0. ms
64 bytes from 129.144.50.21: icmp_seq=2. time=0. ms
64 bytes from 129.144.50.21: icmp_seq=3. time=0. ms
.
.
.
----elvis PING Statistics----
4 packets transmitted, 4 packets received, 0% packet loss
round-trip (ms)  min/avg/max = 0/20/80
```

The statistical message appears after you type the interrupt, or timeout is reached. The packet loss statistic indicates whether the host has dropped packets.

The `ifconfig` Command

The `ifconfig` command displays information about the configuration of a network interface that you specify. (Refer to the manual entry for `ifconfig(8)` for complete information regarding this command.)

For example, if you type the following

```
% ifconfig ie0
```

you may receive the following output:

```
ie0: flags=63<UP,BROADCAST,NOTRAILERS,RUNNING>
inet 129.144.50.28 netmask ffffffff broadcast 129.144.50.0
```

This tells you several things about the Ethernet interface. First, the flags section shows that the interface is up, broadcasting is supported, not using “trailer” link level encapsulation, and that the interface is running. Information included on the second line are the internet address of the host you are using, the netmask being currently used, and the broadcast IP address.

If you use `-a` as an interface name, you will obtain information about all relevant interfaces for a given machine.

If you get output that indicates an interface is not running, it might mean there is a problem with that interface.

The `netstat` Command

The `netstat` command generates displays that show network status.

The `netstat(8C)` man page provides full details about the three ways in which you can invoke the command.

The options you might use most frequently to determine network status are `-s`, `-r`, `-i`, and `-rs` used together.

Displaying Per Protocol Statistics

The `-s` option displays per-protocol statistics for the UDP, TCP, ICMP, and IP protocols. When you run `netstat -s`, the result is the following:

```
udp:
  0 incomplete headers
  0 bad data length fields
  0 bad checksums
  0 socket overflows
tcp:
  165496 packets sent
    139708 data packets (8071601 bytes)
    221 data packets (51939 bytes) retransmitted
  .
  .
  .
  215939 packets received
    136558 acks (for 8077537 bytes)
    834 duplicate acks
    0 acks for unsent data
    126776 packets (4847251 bytes) received in-sequence
  .
  .
  .
  553 connection requests
  516 connection accepts
  1032 connections established (including accepts)
  1124 connections closed (including 0 drops)
  38 embryonic connections dropped
  134488 segments updated rtt (of 135247 attempts)
  327 retransmit timeouts
    0 connections dropped by rexmit timeout
  3 persist timeouts
  33 keepalive timeouts
    0 keepalive probes sent
    33 connections dropped by keepalive
icmp:
  888 calls to icmp_error
  0 errors not generated 'cuz old message too short
  0 errors not generated 'cuz old message was icmp
  .
  .
  .
ip:
  1246461 total packets received
  0 bad header checksums
  0 with size smaller than minimum
  .
  .
  .
```

The statistical information can indicate areas where a protocol is having problems. For example, statistical information from ICMP can indicate problems.

Displaying Communications Controller Status

The `-i` option of `netstat` shows the state of the network interfaces that are configured with your machine. Here is a sample display produced by `netstat -i`.

Name	Mtu	Network	Address	Ipkts	Ierrs	Opkts	Oerrs	Collis
ie0	1500	mtnview-en	speed	14093893	8492	10174659	1119	2314178
ie1	1500	mtnview-en	speed-ebb	9299762	54423	12451748	0	775125

Using this display, you can find out how many packets a machine has transmitted and received on each interface. A machine with active network traffic should show both `Ipkts` and `Opkts` continually increasing.

Displaying Routing Table Status

The `-r` option of `netstat` displays the IP routing table. Here is a sample display produced by `netstat -r` run on machine `ballet`.

```

Routing tables
Destination      Gateway      Flags    Refcnt  Use   Interface
temp8milptp     elvis       UGH      0       0     ie0
irmcpebl-ptp0   elvis       UGH      0       0     ie0
route93-ptp0    speed       UGH      0       0     ie0
mtvb9-ptp0      speed       UGH      0       0     ie0
.
.
.
mtnside         speed       UG       1       567   ie0
ray-net         speed       UG       0       0     ie0
eb5-161         willow      UG       0       0     ie0
eng-bb          willow      UG       0       324   ie0
swan-south      willow      UG       0       0     ie0

```

The first column shows the destination network, the second the router through which packets are forwarded. The `U` flag indicates that the route is up; the `G` flag indicates that the route is to a gateway. The `H` flag indicates that the destination is a fully qualified host address, rather than a network.

The `Refcnt` column shows the number of connections bound to each route, and the `Use` column shows the number of packets sent per route. Finally, the `Interface` column shows the network interface that the route uses.

Displaying Routing Statistics

Combining the options `-rs` with `netstat` produces routing statistics. You can use the resulting display to help determine if your network is having routing problems. Here is sample output from `netstat -rs`

```

routing:
  0 bad routing redirects
  0 dynamically created routes
  0 new gateways due to redirects
  2 destinations found unreachable
 2330 uses of a wildcard route

```

If the output indicates that bad routing redirects occurred or that a number of

destinations were found unreachable, this is indicative of routing problems on your network.

Software Checks

If your network is having problems, here are some actions you can take to diagnose and fix software-related problems.

1. Check the `hosts` database to make sure that the entries are correct and up-to-date. Check the Ethernet addresses in the `ethers` database to make sure that the entries are correct and up to date.
2. Make sure that each network interface in your machine has one `/etc/hostname.??#` file, and that the hostname in it is the correct one.
3. Try to `telnet` (or `rlogin`) to `localhost`. Then try to `telnet` (or `rlogin`) to your machine; if your machine has more than one network interface, try it using the hostname of each interface.
4. Make sure the network daemon `inetd` is running. Type the following:

```
% ps ax | grep inetd
```

The resulting display should resemble the following if the `inetd` daemon is running.

```
136 ? IW 0:13 inetd
1587 p1 S 0:00 grep inetd
```

5. Check the kernel configuration file to make sure the network interfaces have been included.
6. Use the `netstat` command to determine network status as described previously.

`rwhod` and `routed`

The `rwhod` daemon (in `.rwhod`) is not run by default because it has a severe impact on performance. You are strongly advised not to run it. If you need to run it, uncomment its startup line in `/etc/rc`. (Refer to `rwhod(8c)` for more information.)

Normally, `routed`, the routing daemon, runs on each host, and maintains the routing table that enables your machine to pick the best path for sending packets to external networks. `routed` consumes a small amount of memory and CPU time to keep accurate information about the topology of the network.

Whether or not to run `routed` depends on your system configuration. If your machine is memory limited, you may wish to use one of the following configurations:

- If there are no routers on your network then you do not need to run `routed`. You can disable the daemon by removing (or commenting out by inserting a pound sign (#) before each line) the following lines in your `/etc/rc.local` file:


```

if [ ! "$server" ]; then
    if [ -f /usr/etc/in.routed ]; then
        in.routed;                echo -n ' routed'
    fi
    echo '.'
fi

```

- If you have one router in your network, then you can run `routed` only on the router, and disable it on all other machines. All machines on the network route traffic for other networks through this router. To do this, edit the `/etc/rc.local` files on all machines except the router (`router_name`, in the example). Find the following line:

```
echo -n 'starting local daemons'
```

Insert the following two lines just before it:

```

/usr/etc/route add default router_name 1
echo ' /usr/etc/route add default router_name 1'>/dev/console

```

Then, find the following lines and comment them out (insert a '#' before each line) or remove them:

```

if [ ! "$server" ]; then
    if [ -f /usr/etc/in.routed ]; then
        in.routed;                echo -n ' routed'
    fi
    echo '.'
fi

```

Netbooted machines (for instance, diskless workstations) already have a default route, and by default do not run `routed` in Release 4.1, so you should not have to do anything about them.

Logging Network Problems

If you suspect a routing daemon malfunction, you may log its actions. To create a log file of routing daemon actions, just supply a file name when you start up the `in.routed` daemon, for example:

```
# /usr/etc/in.routed /etc/routerlog
```

(Refer to the `routed(8c)` man page for more information about `in.routed`.)

CAUTION: On a busy network this generates almost constant output.

Whenever a route is added, deleted, or modified, a log of the action and a history of the previous packets sent and received will be printed in the log file. To force full packet tracing, specify the `-t` option when the daemon is started up.

The Sun Network File System Service

This chapter explains how to administer the Sun Network File System (NFS[™]) service. It also tells you how to diagnose and fix NFS-related problems. The text explains:

- How NFS works, as an overview
- How to administer an NFS file server
- How to administer an NFS client
- How to set up network security
- How to troubleshoot NFS-related problems

What is NFS

RPC (Remote Procedure Call) procedures provide the means by which one process (termed the *caller process*) can have some other process (the *server process*) execute a procedure call as if the caller process itself had executed the procedure.

NFS is an RPC service that enables machines to share files across a network. In conjunction with other RPC services such as `mount(3R)`, it permits the user to access remote files and hierarchies transparently, as if they were local to the user's machine. To the user, all files accessed through NFS look just like local files. There is no apparent difference between reading and writing a file on a local disk and a file located on the disk of a machine in a different location.

The advantages to using NFS are many. NFS allows multiple machines to use the same files, so data can be made accessible to any machine on a network. Furthermore, storage costs may be kept down by having machines share large programs with each other. You can also achieve consistency in the use of databases, as well as reliability, by having all users read the same set of files.

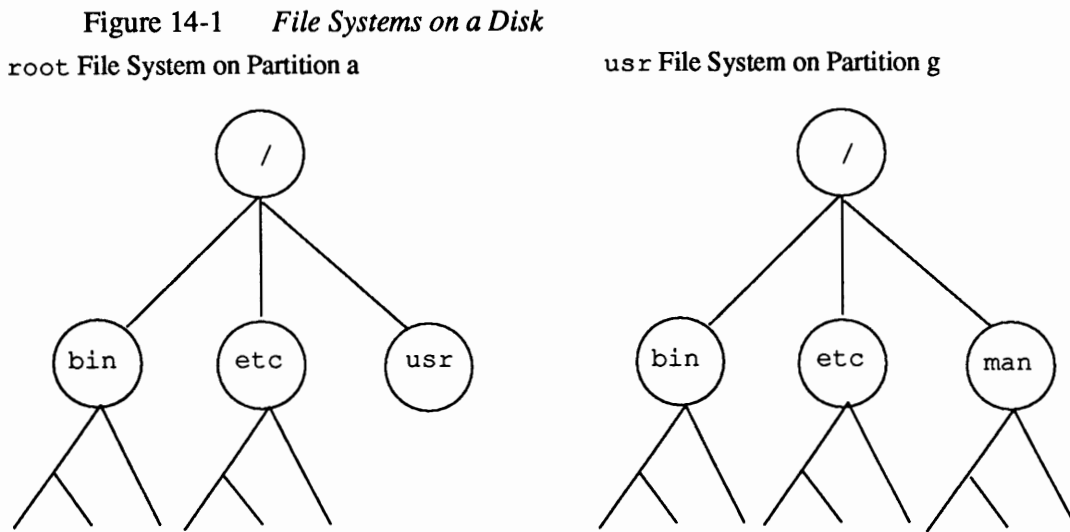
You need to read at least part of this chapter, unless you administer a non-networked, timesharing or stand-alone machine. If you administer an NFS client, you should read the NFS overview and sections that explain how to mount hierarchies, set up network security, and diagnose problems on an NFS client. If you administer an NFS server, read the entire chapter. You may have to fix problems on clients as well as on servers.

14.1. How NFS Works-an Overview

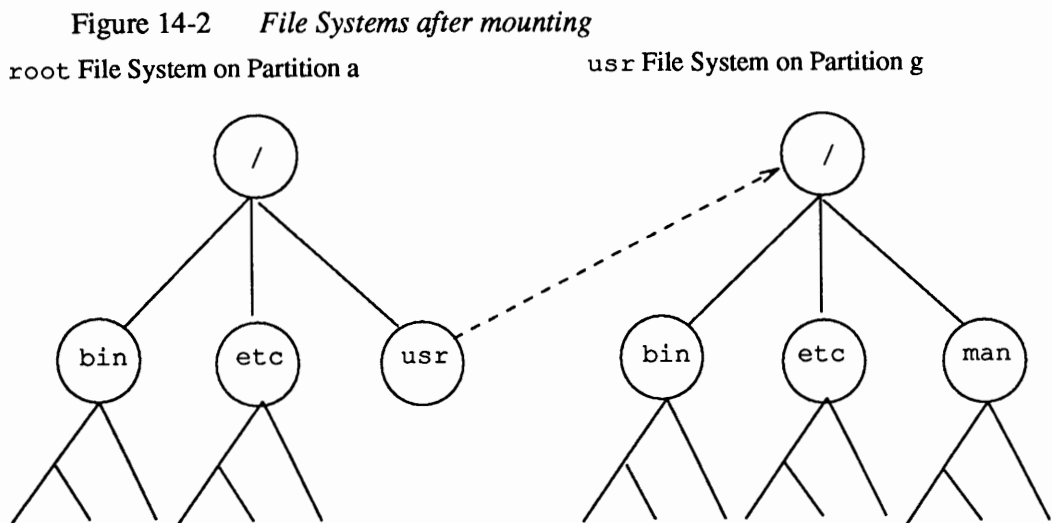
This overview explains how NFS works, including the procedures to share file hierarchies, and how to access those hierarchies on remote machines. It also explains how various daemons (the server processes) interact to enable sharing and mounting. These daemons communicate with the kernel and daemons on other machines to get the job done; they are all based on RPC procedures.

File Hierarchies

When you mount a file system from a local disk onto a mount point, you mount the entire file system, starting at its root. You start, for instance, with two file systems, `root` and `usr`, in different partitions of the disk, as illustrated by the following figure:

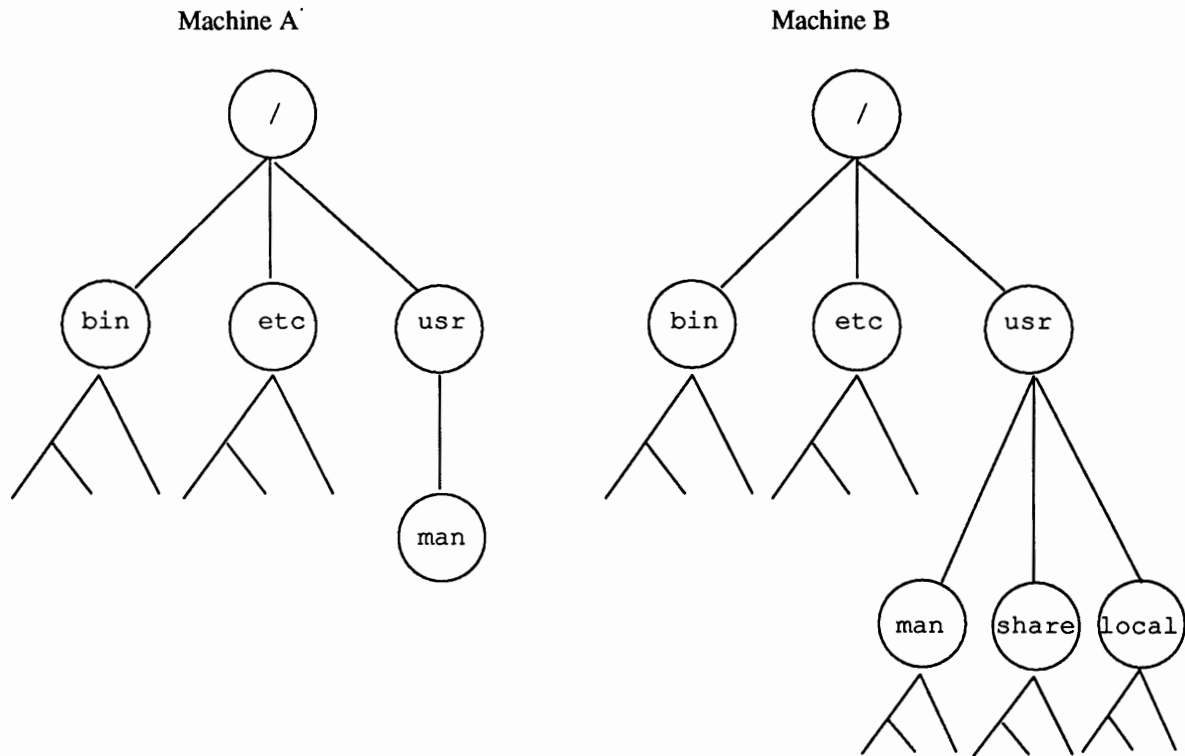


When you mount the `usr` file system on the `/usr` directory of the `root` file system, you access the root of the `usr` partition through the `/usr` directory, as illustrated in the following figure:

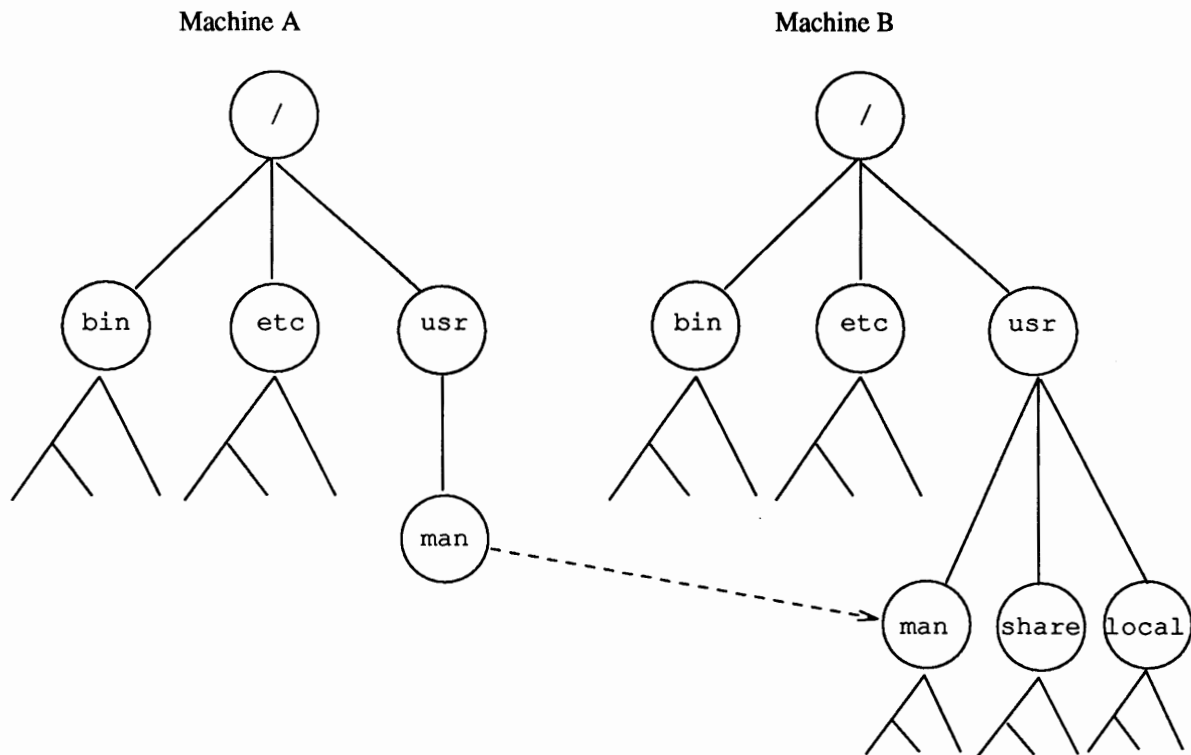


The situation is different in SunOS 4.x when you mount through NFS: you are not constrained to do the mounting starting at the root of the file system. For instance, consider the following two machines, with the following file systems in their disks:

Figure 14-3 *File Systems on Separate Machines*



You do not have to mount the whole `/usr` file system from machine B in order to reach its `/usr/man` directory (including its files and subdirectories). You can simply mount machine B's `/usr/man` as follows:

Figure 14-4 *File Hierarchies on Separate Machines after an NFS Mount*

On UNIX machines this situation coincides with the concept of file systems and portions of file systems; NFS, however, works across operating systems, and the concept of file systems may be meaningless in other, non-UNIX operating systems. The concept of file hierarchies, however, is operating system independent.

Servers and Clients

A *file server* is a machine that shares — that is, lets other machines mount — at least one of its file hierarchies. A *client* is a machine that mounts at least one file hierarchy from a file server. Any machine (except diskless machines, which cannot be servers) can be an NFS server or an NFS client or both at the same time.

Servers and Exporting

On a server, the `exportfs` system program and the `rpc.mountd` and `nfsd` daemons control NFS. When an NFS file server *exports* file hierarchies, it essentially advertises the directories on its disk(s) that it will permit other computers to access. The file server's `/etc/exports` file lists these available directories, which clients are allowed to access them, and any access restrictions that must be applied. When you boot an NFS file server, the `/etc/rc.local` script automatically starts up the `exportfs` program. This program then looks at the `/etc/exports` file and informs the server's kernel about the permissions applicable to each exported file hierarchy.

You can also explicitly export and unexport a file hierarchy after the server is up, or change the export permissions of an exported file hierarchy, by using the `exportfs` command. Explicit exporting is explained in the section *Setting Up and Maintaining an NFS Server*.

Clients and Mounting

Clients access files on the server by *mounting* the server's exported directories. When a client mounts a directory, it does not make a copy of that directory. Rather, the mounting process uses a series of remote procedure calls to enable a client to transparently access the directories on the server's disk.

RPCs running on the file server receive the information in a special format, called XDR format. The information is processed by the `rpc.mountd` daemon, and the client is permitted (or not) to mount the file hierarchy.

Once a client mounts a remote directory, it appears to the users that all they have to do to go to a mounted directory is to type `cd` and the directory's pathname, just as they would for a local directory.

The process by which a client locates a file server that exports the information it wants, then sets up communication between itself and that server, is called *binding*. NFS binding occurs during an NFS mount.

Note: This chapter refers to mounting directories or file hierarchies, but you can also mount single files and access them as any other file, except that you cannot move or remove them.

A client can mount a directory when it boots, explicitly mount a directory when a user issues a `mount` command, or mount a directory through the automounter. The `/etc/fstab` file, first described in the chapter *Introducing the SunOS Operating System*, lists all file hierarchies that the client mounts at boot time. You can also explicitly mount a file hierarchy during a work session by using the `mount` and `umount` commands. Finally, you can enable the automounter so that clients can automatically mount a file hierarchy without having to issue the `mount` command. For further information on the automounter, see the chapter *Using the NFS Automounter*.

14.2. Setting Up and Maintaining an NFS Server

This section explains the files you need to edit or create to set up and successfully maintain an NFS file server. It also describes other actions to take after the server's initial installation.

Setting Up a Server

Recall that when you set up the server during SunInstall, you did the following:

- Identified the machine as an NFS file server.
- Defined the server's disk or disks, indicating which partitions would hold the file systems to be shared among the clients.
- Defined parameters for each of the server's clients on the Client Form.

After you reconfigure the kernel and boot up your NFS server, it has dedicated root directories, home directories, and swap space for each client on its disk, as defined through SunInstall. Furthermore, it has a default `/etc/exports` file that automatically exports these directories, which clients need as soon as they boot up.

The above establishes the machine as an NFS file server for diskless or standalone machines. If you want the machine to export only selected file hierarchies, without exporting the above mentioned dedicated directories, you don't have to set it as a server through SunInstall. All you have to do is to create an appropriate `/etc/exports` file, as explained below, and reboot the machine. If rebooting the machine is not convenient, you can enter the following commands:

```
# exportfs -a
# nfsd 8 &
```

These commands advertise all of the file hierarchies listed in `/etc/exports` and then status up eight `nfsd` server daemons.

If you are running C2 security, you should make sure that port checking is enabled, as explained later in this chapter (or consult your `/etc/rc.local` to see how it's done). For a full discussion on the subject of C2 security, please refer to the chapter on *Administering C2 Security*.

Maintaining a Server

At this point you have only established basic NFS file service. You still need to perform many other NFS-related activities, mostly in the area of file creation and modification. This is particularly important if you do not plan to provide NIS service for your network domain.

Other activities you may need to perform include:

- Adding additional entries to the default `/etc/exports` file, if needed.
- Changing access permissions to `/etc/exports`, as needed, ensuring that all users can read it but only root can write it.
- Directly exporting and unexporting directories during a server's work session.
- If you have more than one NFS server in your network, you may decide that one or more of them will also be clients of the other servers. This is not a recommended strategy; servers should not mount hierarchies from other servers. However, if you must set up a server also as a client, follow the procedures specified below for setting up and maintaining a client.
- Creating mount points in the server's file hierarchies for directories that it will mount from other servers, OR create automount maps, as described in *Using the NFS Automounter*.
- Upgrading a homogeneous server to heterogeneous, as required by your site.
- Upgrading a standalone or time-sharing machine to NFS server, if required.
- Editing the following network databases if you are not going to use NIS:
 - `/etc/passwd`
 - `/etc/group`
 - `/etc/hosts`
 - `/etc/ethers`
- Checking network status on a regular basis, and update related files, as necessary.
- Setting up security for your network, as determined by your site's requirements.
- Diagnosing and fixing NFS related problems as they arise.

Exporting Directories

You can control how a server exports directories in two ways:

- At boot time, through the `/etc/exports` file.
- As needed, by using the `exportfs` command.

The `/etc/exports` File

The `/etc/exports` file lists all file hierarchies that an NFS file server exports to its clients. Mounting itself is initiated by the client. Through `/etc/exports`, servers can control which client(s) may mount a hierarchy by limiting access to it to a desired client or netgroup.

A typical `/etc/exports` on a server for diskless clients, set up through SunInstall, resembles the following:

```

/usr                -access=engineering:accounting
/home              -access=engineering:accounting
/var/spool/mail    -access=engineering:accounting
/export/exec/sun3  -access=engineering
/export/exec/sun3.sunos.4.1 -access=engineering:accounting
/export/exec/kvm/sun3.sunos.4.1 -access=engineering:accounting
/export/root/birch -access=birch,root=birch
/export/swap/birch -access=birch,root=birch
/export/root/oak   -access=oak,root=oak
/export/swap/oak   -access=oak,root=oak
/export/root/willow -access=willow,root=willow
/export/swap/willow -access=willow,root=willow
/export/root/pine  -access=pine,root=pine
/export/swap/pine  -access=pine,root=pine

```

Each line in the file has the syntax:

```
directory -option[,option]
```

where *directory* is the pathname of a directory, a file, or an entire file system or file hierarchy. *option* represents a list of options that indicates which clients are allowed to access the directory, and with what restrictions, if any.

The *SunOS Reference Manual* fully describes all options in the man page for `exports(5)`. The most frequently used options are defined below.

`ro` This option specifies that clients are limited to read-only access to the specified directory. Therefore, they can read the directory's files but cannot change them. If `ro` is not specified for a directory, clients are given read-write (`rw`) access by default.

`root=hostnames` This option indicates that root access to the directory is given only to superusers from a specified *hostname*. In the previous example, the line

```
/export/root/birch -root=birch
```

indicates that root access to `/export/root/birch`

is given only to those who can run superuser on client birch. Note above that root access to the other default client directory, `/export/swap/birch`, is also given to superusers on client birch. A superuser on client oak, for example, instead of accessing files on `/export/root/birch` as user root would access them as user nobody (see below, *Making the Network More Secure.*)

`access=client`

This option indicates that access to the particular directory should be given to the named client, clients, or netgroup. To limit access to a directory to a single client, use the parameter `-access=client_name`. The parameter

```
-access=birch
```

grants mount access to `/export/root/birch` and `/export/swap/birch` to all users on client birch.

You can grant access to more than one client, using a colon-separated list, as follows:

```
/ -access=client1:client2:client3
```

Finally, you can limit mount access to a directory to a designated netgroup if you are running NIS. A netgroup is a network-wide group allowed access to certain network resources for security and organizational reasons. You set up netgroups by modifying the `netgroup` NIS database as described in *The Network Information Service*. See `netgroup(5)` in the *SunOS Reference Manual*.

`secure`

This option indicates that a more secure protocol is used for NFS transactions. See the section *Secure NFS* later in this chapter.

Editing `/etc/exports`

When you bring up your NFS file server after installing Release 4.1, you may want to modify the existing contents of `/etc/exports`. For example, you might want to designate `/usr` as read-write, permit access of a directory to a netgroup, or add more directories for exporting. Once you have modified `/etc/exports`, the NFS file server will automatically export these files every time you boot it up. Therefore, you should modify `/etc/exports` so that it contains directories that you want the server to export at all times.

The following instructions explain how to edit `/etc/exports`:

1. Become superuser and edit the `/etc/exports` file, using your preferred text editor.
2. Enter the full pathname of the file hierarchy you want to export. For example, suppose you want all members of the accounting netgroup to mount a new billing program that you have installed in its own file hierarchy,

/billing. Because this information is confidential, you do not want network users who aren't in the accounting netgroup to mount this directory. The entry in /etc/exports should look like the following:

```
/billing -access=accounting
```

(This example assumes that there is a netgroup called "accounting".)

3. Save the modified file.
4. Run `exportfs` as follows, to update the exports information in the server's kernel.

```
# /usr/etc/exportfs -a
```

The `-a` option tells `exportfs` to send all information in /etc/exports to the kernel.

You cannot export a directory that is either a parent or subdirectory of a directory *within the same file system* that is already exported. It is illegal, for example, to export both /usr and /usr/local if both directories reside on the same disk partition.

Explicitly Exporting Directories with `exportfs`

As mentioned above, you can use the `exportfs` command to explicitly export hierarchies that have not been exported at boot time. You probably will want to use `exportfs` in this fashion for hierarchies that you only want to export for a finite amount of time, then `unexport`.

Note: You must become superuser to use `exportfs` with arguments.

`exportfs` has the following syntax:

```
/usr/etc/exportfs [ -avu ] [ -o options ] [ directory ]
```

The `exportfs(8)` entry in the *SunOS Reference Manual* completely describes this command's arguments. The more commonly used arguments are defined below:

- a Export all the directories in /etc/exports, as illustrated in the previous subsection.
- u Unexport the indicated directories.
- o options Execute the list of options following -o. The arguments to *options* are the same as the options for /etc/exports. For example, you can specify

```
/usr/etc/exportfs -o ro directory
```

to export a directory with read-only access to everybody. You can also use the

```
root=hostname
```

and

```
access=client
```

or

```
access=netgroup
```

parameters in the same fashion as in `/etc/exports`.

directory This is the full pathname of the directory that you want to export or unexport.

To export the `/usr` hierarchy read/write to the netgroup “engineering,” you simply enter the command:

```
# exportfs -o access=engineering /usr
```

If you want to make sure that client “dancer” can access `/usr` read-only, you enter:

```
# exportfs -o access=engineering,ro=dancer /usr
```

You can also use `exportfs` to modify the current list of characteristics of an already exported file hierarchy. For instance, if `/etc/exports` lists `/usr` as exported to the group “engineering,” and there is one or more machines that do not belong to that group but should be able to mount that directory at a particular time, you can use `exportfs` from the command line to accomplish this:

```
# exportfs -o access=engineering:others /usr
```

If you type `/usr/etc/exportfs` without an argument, you receive a display of currently exported directories. You do not have to be superuser in this case.

Unexporting Hierarchies

Unexporting a file hierarchy means making that file hierarchy unavailable to other machines for mounting.

Explicitly Unexporting Hierarchies with `exportfs`

File hierarchies that you export either through the `/etc/exports` file or through an explicit call to `exportfs` can be unexported at any time by using the command `exportfs -u`. For instance, if you want to deny NFS access to the `/usr` file hierarchy, you enter the command:

```
# exportfs -u /usr
```

To unexport all exported hierarchies, enter

```
# exportfs -ua
```

The /etc/xtab File

Whenever `exportfs` (or `exportfs -u`) runs, it updates the `/etc/xtab` file, which lists currently exported directories in a format identical to that of `/etc/exports`. Its contents change whenever you run `exportfs`. On the other hand, `/etc/exports`'s contents remain static until you manually change them. Thus, only `xtab` contains a correct list of the hierarchies being exported at any given time.

Upgrading an NFS Server from Homogeneous to Heterogeneous

As your site's needs grow, you may need to add clients of a different architecture to a network served by a homogeneous NFS server. Only clients with the same application architecture as the server can use the executables in its `/usr` file system. Moreover, machines with the same application architecture but different kernel architecture from the server cannot use its critical `/usr/kvm` files. If your NFS server now must support clients of several application or kernel architectures, you have to install the `/usr` and possibly `/usr/kvm` executables for those architectures from the 4.1 Release tape. For a thorough discussion of kernel and application architectures, and their implications for exporting file systems to clients, see the chapter *Introducing the SunOS Operating System*.

14.3. Setting Up and Maintaining an NFS Client

This section contains information related to setting up and maintaining any type of NFS client machine on a network. You should read this if you are responsible for administering your client machine, or if you are administering both servers and clients on your network.

This section first presents an overview of the actions you must take to set up a client, followed by an overview of any other actions you must take after its initial installation. Finally, it explains in detail of the files and commands involved in the above actions.

Setting Up a Client

The initial activities you need to perform in order to set up a machine as an NFS client include:

- Declaring the machine a diskless or dataless client during `SunInstall`.
- Modifying the client's `/etc/fstab` file to include other hierarchies the client is to mount.
- Making mount points in your client's directory tree for the hierarchies mounted through the `/etc/fstab` file.
- Optionally enabling the automounter for the client, instead of performing the tasks above for hierarchies not related to booting, as explained in *Using the NFS Automounter*.
- After modifying `fstab` and creating the mount points, the client is ready to start mounting the hierarchies as explained below.

Maintaining a Client

Once you set up a machine as an NFS client, you may need to:

- Directly mount and unmount hierarchies as needed.
- Add/remove entries in the client's `/etc/fstab` file.

- Add/remove mount points.
- Make sure that the server is exporting the file hierarchies the client wants to mount.
- Diagnosing and fixing network-related problems on the client.

Mounting Files from an NFS Server

Once an NFS file server exports a file hierarchy, the latter can be accessed by a client through `mount`'s RPC requests. Mounting generally occurs at boot time, when the `mount` program reads the `fstab` file. You can also dynamically mount and unmount hierarchies at any time by using the `mount` and `umount` commands. These two operations can be bypassed by enabling the automounter for NFS mounting (see *Using the NFS Automounter*.)

The next section explains how to set up clients to perform NFS mounts through `/etc/fstab` and the `mount` command.

An NFS Client's `fstab` File

When you first install Release 4.1 on a server, SunInstall creates default `fstab` files for each of the server's clients. The default `/etc/fstab` file for a diskless client resembles the following:

```
oak:/export/root/boomer / nfs rw 0 0
oak:/export/exec/sun3    /usr nfs ro 0 0
oak:/export/exec/kvm/sun3 /usr/kvm nfs ro 0 0
oak:/usr/share          /usr/share nfs ro 0 0
oak:/home/oak /home/oak nfs rw,bg 0 0
```

NFS entries in the file `/etc/fstab` have the following syntax:

```
filesystem directory type options freq pass
```

Note: If you decide to enable the automounter, do not remove these mounts (except `/home`) from the client's `/etc/fstab` file. The client cannot automount these critical directories.

filesystem In the case of an NFS entry, this has the form *host:pathname*, where *host* is the name of the server exporting the directory the client wants to mount, and *pathname* is the name of the directory to be mounted from the indicated server. Note that in the default `fstab` file for the client, the only mounted directories are those that the client requires to operate: its individual `root`, `home`, `share` and `exec` (`/usr` and `/usr/kvm` for the client's architecture). These file hierarchies, along with the client `root` files contain all essential SunOS programs.

directory This is the full pathname of the mount point on the client where *filesystem* should be mounted.

Note: If an NFS file server is also a client of another NFS server, edit the server's `fstab` to include both 4.2 and `nfs` mounts.

type This is the type of mount taking place. In the example above, this is an NFS mount. If the client has a local disk, you can also set up its `fstab` file so that it can perform 4.2 mounts of file systems on its local disk.

- options* This can be any one of a number of options provided for `nfs` or `4.2` mounts. Refer to the `fstab(5)` man page for a complete list. For example, if you are running secure NFS, you may want to select the `secure` option for NFS mounts.
- The example file shows the client performing NFS mounts with the common options `rw` or `ro`. The client can mount its own `root`, `swap`, and `home` directories with read-write access. However, it accesses the server's `/usr` directory with read-only permission; thus, a user on the client cannot add or modify a file in `/usr`.
- freq* This is the interval in days between dumps of the directory. Since these directories are backed up on the server, not on the client, this field should always be zero when the mount type is `nfs`.
- pass* This indicates the `fsck` pass in which the listed file system is checked. For NFS mounts, this should be zero, so no `fsck` checking is attempted.

The contents of `/etc/fstab` remain the same until you change them.

Making Mount Points

Mount points are locations within a directory tree through which your computer accesses mounted hierarchies. Your machine can mount these hierarchies locally from a disk or tape, or remotely from another computer on the network. Any directory can serve as a mount point. The mount points for `/`, `/usr`, and `/home` are provided automatically for you at installation time, together with an extra mount point arbitrarily called `/mnt`.

If you need further mount points, simply create new directories with the `mkdir` command. Just type:

```
# mkdir mount_point
```

where *mount_point* is the pathname of the directory you want to create. Next, make sure the mode and permissions of the new mount point match those of the directory to be mounted on it. Use `chmod` as needed (see the man page for `chmod(1V)`).

Mounting and Unmounting File Hierarchies

This subsection shows how to perform two different mount procedures: statically, through the `fstab` file, or explicitly, through the `mount` and `umount` commands.

Procedures for Modifying the Client's `fstab`

If you want the client always to mount a particular file hierarchies, you should add an entry to its `fstab` file. Follow these directions:

Note: If you edit a client's `fstab` file while logged in to its server, this file is located by default in `/export/root/client/etc/fstab`.

1. Log in as superuser.
2. Edit `/etc/fstab`. Create an entry in the file, using the syntax shown in the previous subsection. List the server exporting the directory you wish to mount, that directory's pathname, the mount point on the client, and so on.
3. Create the mount point in the client's directory tree using the `mkdir` command.
4. Type the following:

```
# mount -a
```

to mount everything in the current `fstab` file.

After you finish these steps, the client will always mount the hierarchies listed in its `/etc/fstab` file until you modify it again.

Explicitly Mounting Hierarchies with `mount`

You can use the `mount` command during the course of client operations to mount a hierarchy. It is advisable to use this command for hierarchies that you only want to access for a finite amount of time. You must be super user to use `mount`.

`mount` explicitly mounts a file hierarchy. It only requires that the client can reach the hierarchy's server over the network, and that the server is exporting it (or a hierarchy above it) to the client.

Here is a form of the `mount` command that you can use for most mount operations. (Refer to the `mount(8)` man page for the complete syntax.)

```
# mount -t type [-rv] -o [options] server:pathname /mount_point
```

These parameters are explained below.

- | | | | | | |
|----------------------------|--|----------------------|--|----------------------------|---|
| <code>-t type</code> | This is the type of mount you want to perform, <code>nfs</code> or <code>4.2</code> , for example. | | | | |
| <code>[-rv]</code> | These are two options that you can supply with <code>mount</code> . <code>-r</code> specifies that you want to mount the directory read-only, even if it is specified as read/write in the <code>fstab</code> file. <code>-v</code> specifies the verbose option, in which <code>mount</code> displays messages as it operates. | | | | |
| <code>-o options</code> | This is a list of options specified after the <code>-o</code> flag. Some NFS-specific options are: <table border="0" style="margin-left: 2em;"> <tr> <td style="vertical-align: top;"><code>[rw ro]</code></td> <td>Indicates whether the mounting is to be done read-only or read/write. The default is <code>rw</code>.</td> </tr> <tr> <td style="vertical-align: top;"><code>[suid nosuid]</code></td> <td>Indicates whether <code>setuid</code> and <code>setgid</code> bits are to be obeyed or ignored on</td> </tr> </table> | <code>[rw ro]</code> | Indicates whether the mounting is to be done read-only or read/write. The default is <code>rw</code> . | <code>[suid nosuid]</code> | Indicates whether <code>setuid</code> and <code>setgid</code> bits are to be obeyed or ignored on |
| <code>[rw ro]</code> | Indicates whether the mounting is to be done read-only or read/write. The default is <code>rw</code> . | | | | |
| <code>[suid nosuid]</code> | Indicates whether <code>setuid</code> and <code>setgid</code> bits are to be obeyed or ignored on | | | | |

execution, respectively. Every hierarchy mounted `rw` is a good candidate for `nosuid`. The default is `suid`.

[`soft` | `hard`]

When the *hard* option is specified, an NFS request affecting any part of the mounted file hierarchy is issued repeatedly until the request is satisfied or until the maximum number of retries is reached (see below, `retry`). When the *soft* option is specified, the NFS request returns an error if it cannot be accomplished, and quits. The default is `hard`.

[`bg` | `fg`]

When the server does not respond to a mount request, retry in the background or the foreground, respectively. The default is `fg`.

`intr`

Allow keyboards interrupts while the NFS request is being issued (see above, `soft` | `hard`).

`retry=n`

Number of times to retry the mount operation. The default for *n* is 10,000 times.

`timeo=n`

Set the timeout on NFS requests to *n* tenths of a second. The default is 7.

`nocto`

Suppress fresh attributes when opening a file.

The remaining options are fully described in the `mount(8)` man page.

server

This is the name of the server exporting the directory.

pathname

This is the full pathname on the server of the directory you want to mount

mount_point

This is the mount point on the client through which the directory is mounted.

Hierarchies accessed through the `mount` command stay mounted during a work session unless you unmount them with the `umount` command. Moreover, if you reboot the machine, the hierarchy is automatically unmounted and remains so (unless you also edited the `fstab` file to include the `mount`.)

For example, to hard mount (with read/write permissions, `nosuid`, interrupt enabled,) the man pages from remote machine `dancer` to the local directory `/usr/man`, you type either:

```
# mount -o nosuid,intr dancer:/usr/man /usr/man
```

or:

```
# mount -o rw,nosuid,hard,intr dancer:/usr/man /usr/man
```

To mount the hierarchy `/usr/local` from the remote machine `dancer` on the mountpoint `/usr/local/dancer`, with read only and soft mounted, you enter:

```
# mount -o ro,soft dancer:/usr/local /usr/local/dancer
```

Here are some final points to remember about `mount`:

- Make the applicable mount point before issuing the `mount` command.
- Consider soft mounting hierarchies such as the man pages so that if the server exporting them goes down, your client will not hang.
- Use the *hard* option with any file hierarchies you mount read-write.
- Use the *hard* option with any file hierarchies you run executables from, whether mounted read-only or read-write. Otherwise, if the server goes down and the process pagefaults, the page-in will fail because of the soft mount and the program will dump. The reason for the failure may not be immediately obvious in this case.
- Use the `nosuid` option with any file hierarchies you mount read-write, unless you have compelling reasons not to do so.
- Use the *intr* option with any file hierarchy you mount *hard*, so that if the server exporting it goes down you can interrupt the current operation.
- Use the *nocto* option on hierarchies which you do not expect will change (such as your `/usr` or a file hierarchy mounted read-only).

Explicitly Unmounting Hierarchies with `umount`

You use `umount` to explicitly unmount a currently mounted file hierarchy, whether the mounting occurred at boot time or through running `mount`. You must be superuser to use `umount`. A simple form of `umount` is shown below. (Refer to the `umount(8)` man page for the complete syntax of this command.)

```
# umount mount_point
```

mount_point is the pathname on the client where you have mounted the files from the file server. You could specify *server:pathname* instead.

The /etc/mtab File

Whenever you explicitly mount or unmount a file hierarchy, the /etc/mtab file is modified to reflect the list of currently mounted file hierarchies. This file has a format and contents similar to those of the fstab file. Some of the file hierarchies mountable through the automounter, however, are displayed with type *ignore*. You can display /etc/mtab by using the cat or more commands, but you cannot edit it, as you would /etc/fstab.

14.4. Obtaining Information

Release 4.1 contains a variety of tools to help you obtain information about the status of the networked file system services.

exportfs

To list all file hierarchies that a server currently exports, including to whom and with what options, enter the command `exportfs` with no arguments, while logged in to the server. You do not have to be super user to use `exportfs` in this fashion.

showmount

To list all the file hierarchies that a server is currently exporting, without having to log in at the server, enter:

```
% showmount -e server
```

To list all the hierarchies that have been mounted from a particular server, in the format *client:directory*, enter:

```
% showmount -a server
```

To obtain a list of the hierarchies that have been remotely mounted by clients, with no further details, enter:

```
% showmount -d server
```

In the case of the `-a` and `-d` options, the mounts shown may not accurately reflect the current situation, since a client that crashes does not modify the database. `showmount` uses /etc/rmtab (on the server), unless the clients reboots and executes `umount -a`.

mount

To list all the resources currently mounted by a machine, enter the command `mount`. This produces a listing like the following:

```
stale:/export/root/junk on / type nfs (rw)
stale:/export/exec/sun3 on /usr type nfs (ro)
stale:/export/exec/kvm/sun3 on /usr/kvm type nfs (ro)
stale:/usr/share on /usr/share type nfs (ro)
stale:/export/local/sun3 on /usr/local type nfs (ro,soft,bg,noquota)
stale:/export/local/share on /usr/local/share type nfs (ro,soft,bg,noquota)
stale:/export/local/src on /usr/local/src type nfs (ro,soft,bg,noquota)
stale:/home/stale on /home/stale type nfs (rw,bg)
```

To list all NFS resources currently mounted, in the same format of `fstab`, enter `mount -p`. This will produce output similar to:

```

stale:/export/root/junk      /                nfs  rw                0  0
stale:/export/exec/sun3     /usr            nfs  ro,nocto          0  0
stale:/export/exec/kvm/sun3 /usr/kvm        nfs  ro,nocto          0  0
stale:/usr/share            /usr/share      nfs  ro,nocto          0  0
stale:/export/local/sun3    /usr/local      nfs  ro,bg,intr,noquota 0  0
stale:/export/local/share   /usr/local/share nfs  ro,soft,bg,noquota 0  0
stale:/export/local/src     /usr/local/src  nfs  ro,soft,bg,noquota 0  0
stale:/home/stale          /home/stale     nfs  rw,bg             0  0

```

The command

```
# mount -p > /etc/fstab
```

can be used to update `/etc/fstab` if there have been changes produced manually by the use of `mount` or `umount`; **Do not** to use this strategy if `automount` is used to mount some file hierarchies, as it would lead to extreme confusion regarding what hierarchies are to be mounted through `mount` and which through `automount`.

`showfh`

When you receive an error message that refers to a file handle, you can use the `showfh` command to obtain the file name corresponding to the file handle if the server from which it is mounted is running the `showfhd` daemon. For instance, if you see an error message like the following:

```
NFS write error 70 on host server fh 305 1 a000 2990 53659758 a0000 2822 7f8bef77
```

you can obtain the name of the file in question by entering:

```
% showfh server 305 1 a000 2990 53659758 a0000 2822 7f8bef77
```

For more details, please consult `showfh(8c)` and `showfhd(8c)` in the *SunOS Reference Manual*.

14.5. Handling NFS Problems

This section describes typical problems that occur on machines using NFS services. Topics discussed include:

- Strategies for tracking NFS problems
- NFS-related error messages

Before trying to clear NFS problems, it is suggested that you have a summary understanding of some of the issues involved. The information in this section contains enough technical details to give experienced network administrators a thorough picture of what is happening with their machines. If you do not yet have this level of expertise, note that it is not important to understand these daemons, system calls and files fully. However, you should be able to at least recognize their names and functions.

Determining Where NFS Service Has Failed

When tracking down an NFS problem, keep in mind that, like all network services, there are three main points of failure: the server, the client, or the network itself. The strategy outlined below tries to isolate each individual component to find the one that is not working.

Note that it is not critical for you to understand in depth how the daemons mentioned work. You simply need to know that they exist, because you may have to restart them if they stop for any reason. Thus, you should familiarize yourself with the following man pages, if you have not done so already:

```
mount (8)
umount (8)
exportfs (8)
mountd (8C)
nfsd (8)
biод (8)
inetd (8C)
inetd.conf (5)
```

This summary begins as the server and the client boot up and read the file `/etc/rc.local`, the pertinent section of which is listed here:

```
if [ -f /etc/exports ]; then
  > /etc/xtab
  exportfs -a
  nfsd 8 &
  if [ -f /etc/security/passwd.adjunct ]; then
    # Warning! Turning on port checking may deny access to
    # older versions (pre-3.0) of NFS clients.
    rpc.mountd
    echo "nfs_portmon/W1" | adb -w /vmunix /dev/kmem >/dev/null 2>&1
  else
    rpc.mountd -n
  fi
fi
```

Note: This explanation is written especially for advanced system administrators and programmers. You do not have to fully understand the entire mount process to clear problems involving remote mounts.

1. When the server boots up, if the file `/etc/exports` exists, indicating that the machine is a server, `/etc/rc.local` executes `exportfs`, which reads the server's `/etc/exports` file, then tells the kernel which file hierarchies the server can export and what access restrictions—if any—are on these files.
2. The file `/etc/rc.local` also starts the `rpc.mountd` daemon and several `nfsd` daemons (usually about eight). The `rpc.mountd` daemon is called with the `-n` flag, a slightly less secure procedure, unless C2 security is in place.
3. When the client boots up, `/etc/rc.local` starts several (usually four) `biод` daemons.
4. The same file also executes `mount -vat nfs` which reads the client's `fstab` file and mounts all NFS-type files mentioned there in a manner

similar to that described below.

5. When the user enters the command:

```
#mount -o ro,soft dancer:/usr/src /usr/src/dancer.src
```

the `mount` command validates that the user has super user permission and that the mount point is a full pathname.

6. `mount` parses the argument *filesystem* into host `dancer` and remote directory `/usr/src`. Because the argument has the form *server:pathname*, `mount` interprets this to mean that this is an NFS mount.
7. If the network uses the NIS service, `mount` obtains from the local portmapper information on how to contact the local NIS binder daemon `ypbind`, which it then calls to determine which server machine to find the NIS server process on. It then calls the `ypserv` daemon on that machine to get `dancer`'s IP address.
8. `mount` calls the portmapper on `dancer` to get the port number of `dancer`'s `rpc.mountd`.
9. `mount` calls `dancer`'s `rpc.mountd` daemon and passes it `/usr/src`, requesting it to send a file handle (`fhandle`) for the directory.
10. If the network uses the NIS service, `dancer`'s `rpc.mountd` daemon calls the NIS server `ypserv` to expand the host names and `netgroups` in the export list for `/usr/src` and to determine the client's host name based on its IP address.
11. The server's `rpc.mountd` daemon handles the client's mount requests. If the directory `/usr/src` is available to the client (or to the public), the `rpc.mountd` daemon does a `getfh(2)` system call on `/usr/src` to get the `fhandle`, and sends it to the client's mount process.
12. `mount` checks if `/usr/src/dancer.src` is a directory.
13. `mount` does a `mount(2)` system call with the directory `/usr/src/dancer.src` as one of the arguments, as well as a data structure containing, among other things, the `fhandle` and the server address, port number, flags and so on.
14. The client kernel looks up the directory `/usr/src/dancer.src` and, if everything is okay, it ties the file handle to the hierarchy in a mount record. From now on all file system requests to that directory and its subdirectories go through the file handle to server `dancer`.
15. The client's kernel does a `statfs(2)` call to `dancer`'s NFS server (`nfsd`).
16. Mount's `mount(2)` system call returns.
17. `mount` opens `/etc/mstab` and adds an appropriate entry to the end, reflecting the new addition to the list of mounted files.
18. Once the hierarchy is mounted, when the client kernel does a file operation it sends the NFS RPC information to the server, where it is read by one of the

`nfsd` daemons to process the file request.

19. The `nfsd` daemons know how a hierarchy is exported from the information sent to the server's kernel by `exportfs`. These daemons allow the client to access the hierarchy according to its permissions.

Debugging Hints

Below are some general pointers for debugging of NFS problems, followed by a list of possible errors and their probable causes.

The `rpc.mountd` daemon must be present on the file server for a remote mount to succeed. Make sure `rpc.mountd` is available for an RPC call by checking that the following lines in `/etc/rc.local` have not been deleted by mistake:

```
if [ -f /etc/exports ]; then
  > /etc/xtab
  exportfs -a
  nfsd 8 &                               echo -n ' nfsd'
  if [ -f /etc/security/passwd.adjunct ]; then
    # Warning! Turning on port checking may deny access to
    # older versions (pre-3.0) of NFS clients.
    rpc.mountd
    echo "nfs_portmon/W1" | adb -w /vmunix /dev/kmem >/dev/null 2>&1
  else
    rpc.mountd -n
  fi
fi
```

Remote mounts also need some number, typically 8, of `nfsd` daemons to execute on NFS servers. Make sure that the line

```
nfsd 8 &                               echo -n ' nfsd'
```

is present in the fragment above.

You can also enable these daemons without rebooting. Become superuser and type:

```
# nfsd 8
```

The client's `biod` daemons are not necessary for NFS to work, but they do increase performance in a very noticeable manner. Make sure that the following (or similar) lines are present in the client's `/etc/rc.local`:

```
if [ -f /usr/etc/biod ]; then
  biod 4;                               echo -n ' biod'
fi
```

You can also enable these daemons without rebooting. Become superuser and type:

```
# biod 4
```

Server Problems

When the network or NFS file server has problems, programs that access hard mounted remote files will fail differently than those that access soft mounted remote files. Hard mounted remote file hierarchies cause the client's kernel to retry the requests until the file server responds again. Soft mounted remote file hierarchies cause the client's system calls to return an error after trying for a while. When you use `mount` with the `bg` option, it retries the mount in the background if the first mount attempt fails.

When a file hierarchy is hard mounted, a program that tries to access it will hang if the server fails to respond. In this case, the message:

```
NFS server hostname not responding, still trying
```

will appear on the console. When the server finally responds, the message

```
NFS server hostname ok
```

will appear on the console.

A program accessing a soft mounted file hierarchy whose server is dead or not responding may or may not check the return conditions of file hierarchy operations. If it does, it should print an error message; otherwise, you may not see an error message on the terminal. But in any case an error message such as the following should appear on the console

```
. . .hostname server not responding: RPC: Timed out
```

If a client is having NFS trouble, check first to make sure the server is up and running. From a client you can type:

```
% rpcinfo -p server_name
```

to see if the server is up at all. `rpcinfo` should give list of program, version, protocol, and port numbers similar to the following:


```

program vers proto  port
100000    2   tcp    111  portmapper
100000    2   udp    111  portmapper
100004    2   udp    658  ypserv
100004    2   tcp    659  ypserv
100004    1   udp    658  ypserv
100004    1   tcp    659  ypserv
100007    2   tcp   1024  ypbind
100007    2   udp   1027  ypbind
100007    1   tcp   1024  ypbind
100007    1   udp   1027  ypbind
100029    1   udp    660  keyserv
100028    1   tcp    665  yupdated
100028    1   udp    667  yupdated
100003    2   udp   2049  nfs
100005    1   udp    709  mountd
100005    2   udp    709  mountd
100005    1   tcp    712  mountd
100005    2   tcp    712  mountd
.
.
.

```

If that works, you can also use `rpcinfo` to check if the `mountd` daemon is running, type the following:

```
% rpcinfo -u server_name mount
```

The command should respond:

```

program 100005 version 1 ready and waiting
program 100005 version 2 ready and waiting

```

If these fail, try logging in to the NFS file server's console. If the server is up but your machine cannot communicate with it, check the Ethernet connections between your machine and the server. If the server is okay and the network is okay, use `ps` to check your client daemons. You should have a `portmap` and several `biod` daemons running. For example, running `ps` should result in output similar to the following:

```

% ps ax
  PID TT  STAT   TIME COMMAND
    0 ?  D     0:00 swapper
    1 ?  S     0:00 /sbin/init -
    2 ?  D     0:00 pagedaemon
   54 ?  IW    0:01 portmap
   59 ?  IW    0:00 ypbind
   61 ?  IW    0:00 keyser
   71 ?  S     0:14 in.routed
   74 ?  I     0:00 (biod)
   75 ?  I     0:00 (biod)
   76 ?  I     0:00 (biod)
   77 ?  I     0:00 (biod)
      .
      .
      .

```

Clearing Remote Mounting Problems

This section deals with problems related to mounting. If `mount` fails for any reason, check the next subsections for specific details. They are arranged according to where they occur in the mounting sequence and are labeled with the error message you are likely to see.

`mount` can get its parameters explicitly from the command line or from `/etc/fstab`. The example below assumes command line arguments, but the same debugging techniques work if `/etc/fstab` is used in the `mount -a` command.

Keep in mind the interaction of the various players in the `mount` request. If you understand this, the problem descriptions below will make a lot more sense.

Error Messages Related to Remote Mounts

Any step in the remote mounting process can fail—some of them in more than one way. The sections below give detailed descriptions of the failures associated with specific error messages.

- `/etc/mtab: No such file or directory`

The mounted file hierarchy table is kept in the file `/etc/mtab`. This file must exist before `mount` can succeed.

- `mount: ... Block device required`

You probably did not specify the remote machine name in the following command:

```
# mount remote.machine:/usr/src/dancer.src
```

The `mount` command assumes you are doing a local mount unless it sees a colon in the file hierarchy name, or unless the `/etc/fstab` specifies that the file hierarchy is mounted via an NFS mount.

- `mount: ... not found in /etc/fstab`

If you issue the `mount` command with only a directory or file hierarchy name but not both, it looks in `/etc/fstab` for an entry whose file hierarchy or directory field matches the argument. For example, the following command:

```
# /mount /dancer.src
```

searches `/etc/fstab` for a line that has a directory name field of `/dancer.src`. If it finds an entry, such as:

```
dancer:/usr/src /dancer.src nfs rw,hard 0 0
```

it will do the mount as if you had typed:

```
# mount -o rw,suid,hard dancer:/usr/src /dancer.src
```

The default options are read-write, hard, and `suid`.

- `/etc/fstab: No such file or directory`

This message indicates that `mount` tried to look up the directory given it as an argument in the `/etc/fstab` file, but could not find the `/etc/fstab` file. You need to create the file.

- `... invalid argument`

NFS is not configured into the kernel. Reconfigure the kernel following the procedures presented in the chapter *Reconfiguring the System Kernel*, making sure that the `NFSCLIENT` option is set.

- `... not in hosts database`

This message indicates that the host specified to mount is not in the `hosts` database, either `/etc/hosts` or its equivalent NIS map. It could also indicate that the NIS daemon `ybind` has died on the machine.

Check the spelling and the placement of the colon in your `mount` command. If the command is correct, and you only get this message for this host name, check the entry in the `hosts` database. If your network is running NIS, make sure that `ybind` is running by typing:

```
# ps ax | grep ybind
```

Try to `rlogin` to another machine, or use `rcp` to remote copy a file to another machine. If this also fails, your `ybind` daemon is probably dead or hung. If you only get this message for this host name, you should check the `hosts.byname` map on the NIS master server. See *The Network Information Service* for more information about NIS problems.

- `mount: directory path must begin with '/'`

The second argument to `mount` is the path of the directory to be used as a mount point. This must be a full pathname, starting at root (/).

- `mount: ... server not responding: \`
`RPC_PMAP_FAILURE - RPC_TIMED_OUT`

Either the NFS file server the client is trying to mount from is down, or its portmapper is dead or hung. Try rebooting the server to restart the `inetd`, `portmap`, and (if running NIS) `ypbind` daemons. If you can't `rlogin` to the server but it is up, check the client's Ethernet connection by trying to `rlogin` to some other machine. You should also check the server's Ethernet connection.

- `mount: ... server not responding: \`
`RPC_PROG_NOT_REGISTERED`

This means that `mount` got through to the portmapper, but the NFS mount daemon `rpc.mountd` was not registered.

- `mount: ...: No such file or directory`

Either the remote directory or the local directory does not exist. Check the spelling of the directory names. Try to use `ls` on both directories.

- `mount: not in export list for ...`

Your machine name is not in the export list for the file hierarchies you want to mount from the server. You can get a list of the server's exported file hierarchies by running

```
# showmount -e hostname
```

If the file hierarchy you want is not in the list, or your machine name or net-group name is not in the user list for the file hierarchy, log in to the server and check the `/etc/exports` file for the correct file hierarchy entry. A file hierarchy name that appears in the `/etc/exports` file but not in the output from `showmount`, indicates a failure in `mountd`. Either it could not understand that line in the file, or `mountd` could not find the file hierarchy, or the file hierarchy name was not a locally mounted file system. See the `exports(5)` man page for more information. If the `/etc/exports` file looks correct, and your network runs NIS, check the server's `ypbind` daemon. It may be dead or hung.

- `mount: ...: Permission denied`

This message is a generic indication that some authentication failed on the file server. It may be that you are not in the export list (see above), that the server could not recognize your machine (`ypbind` is dead), or that the server does not believe you are who you say you are. Check the file server's `/etc/exports` file, and, if applicable, `ypbind`.

- `mount: ...: Not a directory`

Either the remote path or the local path is not a directory. Check the spelling in your command, and try to run `ls` on both directories.

- Must be root to use `mount`

You have to run `mount` as root on your machine.

- Stale NFS file handle

You usually receive this message when a directory you have successfully mounted is removed from a server or is unexported. As long as your machine mounts a directory, it keeps the same file handle. But if someone removes or unexports the directory from its server, that file handle becomes invalid, or “stale.”

Unmount the directory and try mounting it again.

When `mount` Hangs Without an Error Message

The `mount` command hangs indefinitely if no `nfsd` daemons are running on the NFS server. This happens when no `/etc/exports` file exists on the server when it is booted. To clear the problem, you need to:

1. Become superuser on the NFS server, and create the `/etc/exports` file.
2. Type

```
# exportfs -a
```

to send the information in `/etc/exports` to the kernel

3. Restart the `nfsd` daemons by typing

```
# nfsd 8
```

Fixing Hung Programs

If programs hang doing file-related work, your NFS server may be dead. You may see the following:

```
NFS server hostname not responding, still trying
```

on your console. The message indicates that NFS server *hostname* is down. This indicates a problem with your NFS server or with the Ethernet. Programs can also hang if an NIS server dies.

If your machine hangs completely, check the server(s) from which you have mounted file hierarchies. If one of them (or more) is down, do not be concerned. When the server comes back up, your programs continue automatically. No files are destroyed.

If a soft mounted server dies, your work should not be affected. Programs that time out trying to access soft mounted remote files will fail with a message reported that they have timed out, but you should still be able to access your other file systems.

If all servers are running, ask someone else using the same file servers as your machine if they are having trouble. If more than one machine is having problems getting service, this indicates a problem with the file server. Log in to the file server. Run `ps` to see if `nfsd` is running and accumulating CPU time. (This implies running `ps -ax` a few times, letting some time pass between each call.) If not, you may be able to kill and then restart `nfsd`. If this does not work, you will have to reboot the server.

If other systems seem to be up and running, check your Ethernet connection and the connection of the server.

Fixing a Machine that Hangs Part Way Through Boot

If your machine boots up normally until it tries to do remote mounts, probably one or more servers is down, or your network connection is bad. Try to reboot in single-user mode. Then use the `mount` command to manually mount each directory normally mounted through the `/etc/fstab` file. See the previous two subsections for more help.

Speeding Up Slow Access Times

If access to remote files seems unusually slow, type:

```
# ps aux
```

on the server to be sure that it is not being adversely affected by a runaway daemon, bad `tty` line, and so on. If the server seems okay and others are getting good response, make sure your `biod` daemons are running. Try the following steps:

1. Run

```
# ps ax | grep biod
```

(at the client) and look for `biod` daemons in the display. To determine if the `biods` are hung, `rcp` a large file from a remote system, then run `ps` as above again. If the `biods` do not accumulate any CPU time (as shown by the `TIME` column,) they are probably hung. If they are not running or are hung, continue with these steps.

2. Kill these processes as follows:

```
# kill -9 pid1 pid2 pid3 pid4
```

4. Restart them by typing:

```
# biod 4
```

If the `biods` are okay, check your Ethernet connection. The command `netstat -i` introduced in the chapter *The SunOS Network Environment*, tells you if you are dropping packets. Also, you can use the commands `nfsstat -c` and `nfsstat -s` to tell if the client or file server is doing a lot of retransmitting. A retransmission rate of 5 percent is considered high. Excessive retransmission usually indicates a bad Ethernet board, a bad Ethernet tap, a

mismatch between board and tap, or a mismatch between your Ethernet board and the server's board.

14.6. Making the Network More Secure

To obtain the maximum degree of security available for remote mounts, file hierarchies should be exported and mounted with the `secure` option activated; this is possible only if the NIS service is running. Please consult the chapter on *The Network Information Service* and the section *Secure NFS* later in this chapter.

Allowing Root Access over the Network

Under NFS, a server exports file hierarchies it owns, so that clients can remotely mount them and users logged in on the clients can access them transparently, as if they were local to the client. However, a user who becomes super user at a client is denied root access to a remotely mounted file hierarchy. When a person logged in as `root` on one host requests access to a particular file from NFS, the user ID of the requester is changed to the user ID of the user name `nobody` — generally 65534. The access rights of user `nobody` are the same as those given to the public for a particular file. For example, if the public only has execute permission for a file, then user `nobody` can only execute that file.

When you export a file hierarchy, you can permit `root` on a particular machine to have root access to that hierarchy by specifying it in `/etc/exports` on the server. For example, suppose you wanted `root` at machine `samba` (but no others) to have superuser access to the exported directory `/usr/src`. You would enter the following line in `/etc/exports`:

```
/usr/src -root=samba
```

If you want more than one client to have root access, you can specify a list, as follows:

```
/usr/src -root=samba:raks:jazz
```

You can also enable superuser access for processes with user ID 0 on all clients, again using the `/etc/exports` file. If you wanted to allow superuser access for all client processes with user ID 0 accessing `/usr/src` you would add the following line to `/etc/exports`:

```
/usr/src -anon=0
```

The `anon` is short for “anonymous.” Anonymous requests, by default, get their user ID changed from its previous value (whatever it may be) to 65534, the user ID of `nobody`. NFS servers label as anonymous any request from a root user (someone whose current effective user ID is 0) who is not in the root list in `/etc/exports`. The above sample line in `/etc/exports` tells the kernel to use the value 0 instead of 65534 for anonymous requests. The result is that all root users retain their user ID of 0. Unfortunately, this has the side effect that anonymous users are granted root access. For obvious security reasons, you should weigh very carefully whether to use the `root` and `anon` options. Finally, the following line in `/etc/exports` effectively prohibits anonymous access:

```
/usr/src -anon=-1
```

See `exports(5)` and `exportfs(8)` for more information on exporting directories.

Privileged Ports

The SunOS operating system, and others whose networking is derived from Berkeley UNIX, have some TCP and UDP source ports to which only privileged users can attach. These are known as *privileged ports*. Currently, NFS does not check to see if a client is bound to one of these. That is, an NFS server has no way of knowing whether a client's file request originated from the real client's kernel or from someone's user program. If you get the following error message:

```
NFS request from unprivileged port
```

you should also see the IP address of the client on the console screen of the server. You can turn on server port checking by making the following change:

```
# adb -w /vmunix
nfs_portmon?W1
_nfs_portmon: 0x0 =          0x1
<Control-D>
```

The next time you boot your system, the source ports will be checked. If you can trust all of the root users on your network, then just doing the above is enough. But be warned: most non-UNIX and some UNIX systems do not enforce the privileged port convention (in particular, PCs with 3Com boards). Another warning: all of a server's SunOS clients should be running software release 3.0 or higher for this to work.

If you run C2 security in your network, the above is done automatically on all the servers, in the lines in `/etc/rc.local` that say:

```
if [ -f /etc/security/passwd.adjunct ]; then
    # Warning! Turning on port checking may deny access to
    # older versions (pre-3.0) of NFS clients.
    rpc.mountd
    echo "nfs_portmon/W1" | adb -w /vmunix /dev/kmem >/dev/null 2>&1
else
    rpc.mountd -n
fi
```

The Lock Manager

Sun Release 4.1 supports a System V compatible record and file locking service. The lock manager functionality applies to local record and file locking as well as to record and file locking of remote file hierarchies on the network. File locking is accomplished with the `lockf` and `fcntl` functions. The user, program, or application can optionally invoke the service on a per-transaction, or per-application basis. Once a record is locked by one user or application, another

user or application trying to lock the same record will either (a) receive an error message or (b) be placed in a queue and be appropriately notified when the lock becomes available.

The *Lock Manager* implements a System V Interface Definition (SVID) compatible record and file locking service. When multiple users attempt to access the same record at the same time, only one will have access and the rest will be locked out.

Synchronizing the time

To prevent a client from having a totally different notion of time than its server, diskless clients coordinate their date with that of their server at boot time through the following lines in `/etc/rc.local`:

```
if [ "$server" ]; then
    intr -a rdate $server
fi
```

The time a machine keeps, however, is not absolute and depends on a variety of factors. Sooner or later two machines whose date has been set simultaneously will drift and display different times. To prevent a client from having a totally different notion of time than its server, you may want to issue the command `rdate` periodically on the client, either manually or through the client's `/usr/spool/cron/crontabs/root` file.

Release 4.1 of the SunOS operating system does not supply a mechanism for maintaining the same date value across a network other than this command. You should consider having all machines obtain their date from a single server at boot time and/or through daily invoking of `rdate` through the `crontab` mechanism.

14.7. Secure NFS

Although, as we have seen in this chapter, NFS is a powerful and convenient way for file sharing between heterogeneous machine architectures and operating systems, it is not without its shortcomings.

Security Shortcomings of NFS

NFS servers authenticate a file request by authenticating the machine making the request, but not the user. On NFS-based file hierarchies that are exported to machines where superuser access is not controlled, it is a simple matter of running `su` to impersonate the rightful owner of a file. The familiar command `rlogin` is subject to exactly the same attacks as NFS because it uses the same kind of authentication.

Given `root` access and a good knowledge of network programming, anyone could be capable of injecting arbitrary data into the network, and picking up any data from it. However, on a local area network, no machine is capable of packet smashing – capturing packets before they reach their destination, changing the contents, then sending packets back on their original course – because packets reach all machines, including the server, at the same time. Packet smashing is possible on a gateway, though, so make sure you trust all gateways on the network. The most dangerous attacks are those involving the injection of data, such as impersonating a user by generating the right packets, or recording

conversations and replaying them later. These attacks affect data integrity. Attacks involving passive eavesdropping – merely listening to network traffic without impersonating anybody – are not as dangerous, since data integrity is not compromised. Moreover, making sense of network traffic is not easy, and users can protect the privacy of sensitive information by encrypting data that goes over the network.

An easy solution to network security problems is to leave the solution to each application. A far better solution is to put authentication at the RPC level. The result is a standard authentication system that covers all RPC-based applications, such as NFS and NIS. This system allows the authentication of users as well as machines. The advantage of this is that it makes a network environment more like a time-sharing environment. Users can log in on any machine, just as they could log in on any terminal. Their login password is their passport to network security. No knowledge of the underlying authentication system is required. The system is as secure and easy to use as a time-sharing system.

SunOS 4.1 includes an authentication system that greatly improves the security of network environments. The system is general enough to be used by other UNIX and non-UNIX systems. The system uses DES encryption and public key cryptography to authenticate both users and machines in the network. (DES stands for Data Encryption Standard.)

Public key cryptography is a cipher system that involves two keys: one public and the other private. The public key is published, while the private key is not; the private (or secret) key is used to encrypt and decrypt data. Sun's system differs from some other public key cryptography systems in that the public and secret keys are used to generate a common key, which is used in turn to create a DES key. DES is relatively fast, and, to make it even faster, optional Sun hardware is available at present for most Sun computers.

In order to use this public key cryptography over the network, you must also run NIS — see the chapter on *The Network Information Service* — and you must install the Encryption Kit (available only in the U.S.A.).

14.8. Administering Secure NFS

This section describes what the system administrator must do in order to create a secure file hierarchy over NFS. First, the public and private keys must be generated for each user. Then, file hierarchies must be exported secure, and mounted secure.

1. The `publickey` database contains three fields for each user:

```
netname  user's public key : user's secret key
```

where *netname* may be the name of a user or of a machine, *user's public key* is that key in hexadecimal notation, and *user's secret key* is that key encrypted with the user's password, also in hexadecimal notation. If the *netname* is the name of a user, then the `publickey` entry takes the form `unix.uid@domain`. If it is the name of a machine, it takes the form `unix.host@domain`.

The corresponding NIS maps is available to NIS clients as `publickey.byname`, but the database should reside on the NIS master server in the file `publickey` in the directory `/etc` (or similar).

The program `newkey` is provided to ease the task of updating the database. Become superuser at the master server and invoke `newkey` for a given user:

```
# newkey -u username
```

or for the superuser in a given host machine:

```
# newkey -h hostname
```

and at the prompt enter the appropriate login password. The program will then create a new public/secret key pair in `publickey`, encrypted with the login password of the given user. As normally installed, the only user is `nobody`.

Users can later on modify their own entries by using the program `chkey`. The user types:

```
% chkey
```

and then responds to prompts from the command. A typical `chkey` session would look like this:

```
willow% chkey
Generating new key for username
Password: user enters password
Sending key change request to server...
Done.
willow%
```

If there is an existing entry for `nobody`, users can use the `chkey` program to create their own entries. The administrator can delete the entry for `nobody` in the `publickey` file on the NIS master server to prevent users from creating new entries for themselves, but if they already have one they can still modify it.

Note that NIS takes time to propagate a new map, so it's a good idea for users to run `chkey`, or for the administrator to run `newkey`, just before going home for the night.

Also note that in order for `newkey` and `chkey` to be able to run properly, the daemon `rpc.yupdated` must be running on the master server. For more information, see *The Network Information Service*.

2. Verify that the `keyserv` and `ybind` daemons (which are normally started by `/etc/rc.local`) are still running. `keyserv` performs public key encryption and stores the private key (encrypted, of course) in `/etc/keystore`:

```
% ps aux | grep keyserv
root 61 0.0 0.0 56 0 ? IW Dec 29 0:08 keyserv
```

When users log in with `login` or remote log in with `rlogin`, these programs use the typed password to decrypt the secret key stored in the `publickey.byname` map. This becomes the private key, and gets passed to the `keyserv` daemon. If users don't type a password for `login` or `rlogin`, either because their password field is empty or because their machine is in the `hosts.equiv` file of the remote host, they can still place a private key in `/etc/keystore` by invoking the `keylogin` program. Before logging out, users can use the `keylogout` program to remove their private key. Please refer to `keylogin(1)` and `keylogout(1)` in the *SunOS Reference Manual*.

3. To keep their secret key synchronized with their login password, no user (except root) should have an entry in the local `/etc/passwd` file; it is better to use the default `/etc/passwd` as distributed with SunOS. This way, when users invoke `passwd` this command will behave like `yppasswd` (or `passwd -y`), and it will change the NIS database.
4. When you reinstall, move, or upgrade a machine, save `/etc/keystore` and `/etc/.rootkey` (the latter file contains the private key for root) along with everything else you normally save.

Note that if users use `login`, `rlogin`, or `telnet` to gain access to another machine, are asked for their password, and type it correctly, they have given away access to their own account. This is because their secret key is now stored in `/etc/keystore` on that remote machine. This is only a concern if the remote machine is not trusted. If this is the case, users should never log in to a remote machine if it asks for their password, or should make sure to use the `keylogout` command just before logging out. However, using `keylogout` is also no guarantee that the secret key is actually destroyed – a malicious superuser can install a fake version of `login`, `keylogin`, or `keylogout`. Another alternative is for users to use NFS to remote mount the files they are looking for.

5. Edit the `/etc/exports` file and add the `-secure` option for file hierarchies that should use DES authentication. Here's how a server might export secure file hierarchies:

```
/usr/src    -secure,access=engineering
```

In this example, `engineering` is the only network group with access to the `/usr/src` file hierarchy.

6. For each client machine, edit `/etc/fstab` (or have users edit their own files) to include `secure` as a mount option on each secure file hierarchy. Here's how a client might mount secure file hierarchies:

```
server:/usr/src /usr/src nfs rw,secure,intr,bg 0 0
```

In this example, the `/usr/src` file hierarchy from server `server` is mounted hard, read/write, and secure. If a client machine does not mount a secure file hierarchy as `secure`, everything works OK, except that users have access as nobody (user ID 65534), rather than as themselves.

The remainder of this chapter discusses the theory of secure networking, and is useful as a background for both users and administrators.

14.9. RPC Authentication

Secure RPC is at the core of secure NFS. To understand the big picture, it's necessary to understand how authentication works in RPC. RPC's authentication is open-ended: a variety of authentication systems may be plugged into it and may coexist on the network. Currently there are two: UNIX and DES. UNIX authentication is the older, weaker system; DES authentication is the new system discussed in this chapter.

Two terms are important for any RPC authentication system: *credentials* and *verifiers*. Using ID badges as an example, the credential is what identifies a person: a name, address, birth date, etc. The verifier is the photo attached to the badge: you can be sure the badge has not been stolen by checking the photo on the badge against the person carrying it. In RPC, things are similar. The client process sends both a credential and a verifier to the server with each RPC request. The server sends back only a verifier, since the client already knows the server's credentials.

UNIX Authentication

In UNIX authentication the credentials contain the client's machine-name, user ID, group ID, and group-access-list. The verifier, on the other, contains **nothing!** There are two problems with this system. The glaring problem is the empty verifier, which makes it easy to cook up the right credential using `hostname` and `su`. If you trust all root users in the network, this is not really a problem. But many networks are not this secure. The NFS tries to combat deficiencies in UNIX authentication by checking the source Internet address of `mount` requests as a verifier of the `hostname` field, and accepting requests only from privileged Internet ports. Still, it is not difficult to circumvent these measures, and NFS really has no way to verify the user-ID.

The other problem with UNIX authentication appears in the name UNIX. It is unrealistic to assume that all machines on a network will be UNIX machines. The NFS works with machines running other operating systems, but UNIX authentication breaks down when applied to them. For instance, MS-DOS® doesn't even have a notion of different user IDs.

Given these shortcomings, it is clear what is needed in a different authentication system: operating system independent credentials, and secure verifiers. This is the essence of DES authentication discussed below.

DES Authentication

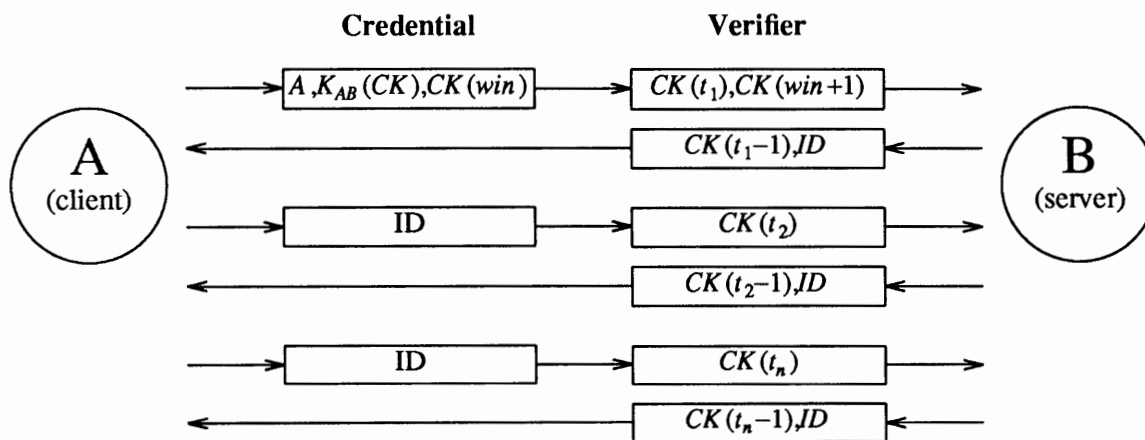
The security of DES authentication is based on a sender's ability to encrypt the current time, which the receiver can then decrypt and check against its own clock. The timestamp is encrypted with DES. Two things are necessary for this scheme to work: 1) the two agents must agree on what the current time is, and 2) the sender and receiver must be using the same encryption key.

If a network has time synchronization (Berkeley's TEMPO for example), then client/server time synchronization is performed automatically. However, if this is not available, timestamps can be computed using the server's time instead of

network time. In order to do this, the client asks the server what time it is, before starting the RPC session, then computes the time difference between its own clock and the server's. This difference is used to offset the client's clock when computing timestamps. If the client and server clocks get out of sync to the point where the server begins rejecting the client's requests, the DES authentication system resynchronizes with the server.

Here is how the client and server arrive at the same encryption key. When a client wishes to talk to a server, it generates at random a key to be used for encrypting the timestamps (among other things). This key is known as the *conversation key*, CK . The client encrypts the conversation key using a public key scheme, and sends it to the server in its first transaction. This key is the only thing that is ever encrypted with public key cryptography. The particular scheme used is described further on in this chapter. For now, suffice to say that for any two agents A and B, there is a DES key K_{AB} that only A and B can deduce. This key is known as the *common key*, K_{AB} .

Figure 14-5 *DES Authentication Protocol*



The figure above illustrates the authentication protocol in more detail, describing client A talking to server B. A term of the form $K(x)$ means x encrypted with the DES key K . Examining the figure, you can see that for its first request, the client's credential contains three things: its name A , the conversation key CK encrypted with the common key K_{AB} , and a thing called *win* (window) encrypted with CK . What the window says to the server, in effect, is this:

I will be sending you many credentials in the future, but there may be crackers sending them too, trying to impersonate me with bogus timestamps. When you receive a timestamp, check to see if your current time is somewhere between the timestamp and the timestamp plus the window. If it's not, please reject the credential.

For secure NFS file hierarchies, the window currently defaults to 60 minutes. The client's verifier in the first request contains the encrypted timestamp and an encrypted verifier of the specified window, $win+1$. The reason this exists is the following. Suppose somebody wanted to impersonate A by writing a program that instead of filling in the encrypted fields of the credential and verifier, just stuffs in random bits. The server will decrypt CK into some random DES key, and

use it to decrypt the window and the timestamp. These will just end up as random numbers. After a few thousand trials, there is a good chance that the random window/timestamp pair will pass the authentication system. The window verifier makes guessing the right credential much more difficult.

After authenticating the client, the server stores four things into a credential table: the client's name A , the conversation key CK , the window, and the timestamp. The reason the server stores the first three things should be clear: it needs them for future use. The reason for storing the timestamp is to protect against replays: the server will only accept timestamps that are chronologically greater than the last one seen, so any replayed transactions are guaranteed to be rejected. The server returns to the client in its verifier an index ID into its credential table, plus the client's timestamp minus one, encrypted by CK . The client knows that only the server could have sent such a verifier, since only the server knows what timestamp the client sent. The reason for subtracting one from it is to insure that it is invalid and cannot be reused as a client verifier.

The first transaction is rather complicated, but after this things go very smoothly. The client just sends its ID and an encrypted timestamp to the server, and the server sends back the client's timestamp minus one, encrypted by CK .

Public Key Encryption

The particular public key encryption scheme Sun uses is the Diffie-Hellman method. The way this algorithm works is to generate a *secret key* SK_A at random and compute a *public key* PK_A using the following formula (PK and SK are 128 bit numbers and α is a well-known constant):

$$PK_A = \alpha^{SK_A}$$

Public key PK_A is stored in a public directory, but secret key SK_A is kept private. Next, PK_B is generated from SK_B in the same manner as above. Now common key K_{AB} can be derived as follows:

$$K_{AB} = PK_B^{SK_A} = (\alpha^{SK_B})^{SK_A} = \alpha^{(SK_A SK_B)}$$

Without knowing the client's secret key, the server can calculate the same common key K_{AB} in a different way, as follows:

$$K_{AB} = PK_A^{SK_B} = (\alpha^{SK_A})^{SK_B} = \alpha^{(SK_A SK_B)}$$

Notice that nobody else but the server and client can calculate K_{AB} , since doing so requires knowing either one secret key or the other. All of this arithmetic is actually computed modulo M , which is another well-known constant. It would seem at first that somebody could guess your secret key by taking the logarithm of your public one, but M is so large that this is a computationally infeasible task. To be secure, K_{AB} has too many bits to be used as a DES key, so 56 bits are extracted from it to form the DES key.

Both the public and the secret keys are stored indexed by netname in the NIS map `publickey.byname`; the secret key is DES-encrypted with the user's login password. When a user logs in to a machine, the `login` program decrypts the encrypted secret key with the login password, and gives it to a secure local keyserver to save for use in future RPC transactions. (This is what `keylogin` does also). Note that ordinary users do not have to be aware of their public and

secret keys. In addition to changing a user's login password, the `yppasswd` program (or `passwd` if the user does not have an entry in the local `/etc/passwd`, or `passwd -y`) randomly generates a new public/secret key pair as well.

The keyserver `keyserv(8c)` is an RPC service local to each machine that performs all of the public key operations, of which there are only three. They are:

```
setsecretkey(secretkey)
encryptsessionkey(servername, des_key)
decryptsessionkey(clientname, des_key)
```

`setsecretkey()` tells the keyserver to store away the users' secret key SK_A for future use; it is normally called by `login`. The client program calls `encryptsessionkey()` to generate the encrypted conversation key that is passed in the first RPC transaction to a server. The keyserver looks up `servername`'s public key and combines it with the client's secret key (set up by a previous `setsecretkey()` call) to generate the key that encrypts `des_key`. The server asks the keyserver to decrypt the conversation key by calling `decryptsessionkey()`.

Note that implicit in these procedures is the name of caller, who must be authenticated in some manner. The keyserver cannot use DES authentication to do this, since it would create a deadlock. The keyserver solves this problem by storing the secret keys by user ID, and only granting requests to local root processes. The client process then executes a `setuid` process, owned by root, which makes the request on the part of the client, telling the keyserver the real user ID of the client. Ideally, the three operations described above would be system calls, and the kernel would talk to the keyserver directly, instead of executing the `setuid` program.

Naming of Network Entities

The old UNIX authentication system has a few problems when it comes to naming. Recall that with UNIX authentication, the name of a network entity is basically the user ID. These user IDs are assigned per NIS naming domain, which typically spans several machines. We have already stated one problem with this system, that it is too UNIX system oriented, but there are two other problems as well. One is the problem of user ID clashes when domains are linked together. The other problem is that the super-user (with user ID of 0) should not be assigned on a per-domain basis, but rather on a per-machine basis. By default, NFS deals with this latter problem in a severe manner: it does not allow root access across the network by user ID 0 at all.

DES authentication corrects these problems by basing naming upon new names called *netnames*. Simply put, a netname is just a string of printable characters, and fundamentally, it is really these netnames that are authenticated. The public and secret keys are stored on a per-netname, rather than per-username, basis. The NIS map `netid.byname` maps the netname into a local user ID, group ID, and group-access-list, though non-Sun environments may map the netname into something else.

The Internet naming problem is solved by choosing globally unique netnames. This is far easier than choosing globally unique user IDs. In the Sun environment, user names are unique within each NIS domain. Netnames are assigned by concatenating the operating system and user ID with the NIS and Internet domain names. For example, a UNIX system user with a user ID of 508 in the domain `eng.sun.com` would be assigned the following netname: `unix.508@eng.sun.com`. A good convention for naming domains is to append the Internet domain name (COM, EDU, GOV, MIL) to the local domain name. Thus, the NIS domain `eng` within the Internet domain `sun.com` becomes `eng.sun.com`.

The problem of multiple super-users per domain is solved by assigning netnames to machines as well as to users. A machine's netname is formed much like a user's. For example, a UNIX machine named `hal` in the same domain as before has the netname `unix.hal@eng.sun.com`. Proper authentication of machines is very important for diskless machines that need full access to their home directories over the net.

Non-Sun environments will have other ways of generating netnames, but this does not preclude them from accessing the secure network services of the Sun environment. To authenticate users from any remote domain, all that has to be done is make entries for them in two NIS databases. One is an entry for their public and secret keys, (in the `publickey.byname` map) the other is for their local user ID and group-access-list mapping (in the `netid.byname` map). Upon doing this, users in the remote domain will be able to access all of the local network services, such as NFS and remote logins.

Applications of DES Authentication

The first application of DES authentication is a generalized NIS update service. This service allows users to update private fields in NIS databases. So far only the NIS map `publickey` employs the DES-based update service.

The second application of DES authentication is the most important: a more secure Network File System. There are three security problems with the old NFS using UNIX authentication. The first is that verification of credentials occurs only at mount time when the client gets from the server a piece of information that is its key to all further requests: the *file handle*. Security can be broken if one can figure out a file handle without contacting the server, perhaps by tapping into the net or by guessing. After an NFS file hierarchy has been mounted, there is no checking of credentials during file requests, which brings up the second problem. If a file hierarchy has been mounted from a server that serves multiple clients (as is typically the case), there is no protection against someone who has root permission on their machine using `su` (or some other means of changing user ID) gaining unauthorized access to other people's files. The third problem with NFS is the severe method it uses to circumvent the problem of not being able to authenticate remote client super-users: denying them super-user access altogether.

The new authentication system corrects all of these problems. Guessing file handles is no longer a problem since in order to gain unauthorized access, the miscreant would also have to guess the right encrypted timestamp to place in the

credential, which is a virtually impossible task. The problem of authenticating root users is solved, since the new system can authenticate machines. At this point, however, secure NFS is not used for root file systems. Root users of non-secure file hierarchies are identified by IP address.

Actually, the level of security associated with each file hierarchy may be altered by the administrator. The file `/etc/exports` contains a list of file hierarchies and which machines may mount them. By default, file hierarchies are exported with UNIX authentication, but the administrator can have them exported with DES authentication by specifying `-secure` on any entry in the `/etc/exports` file. Associated with DES authentication is a parameter: the maximum window size that the server is willing to accept.

Security Issues Remaining

There are several ways to break DES authentication, but using `su` is not one of them. In order to be authenticated, a user's secret key must be stored by the user's workstation. This usually occurs when the user logs in, with the `login` program decrypting the secret key with the login password, and storing it away for the user. If somebody tries to use `su` to impersonate the user, it won't work, because they won't be able to decrypt the secret key. Editing `/etc/passwd` isn't going to help them either, because what they need to edit, the encrypted secret key, is stored in the NIS database. If you log into somebody else's workstation and type in your password, then your secret key would be stored in their workstation and they could use `su` to impersonate you. But this is not a problem since you should not be giving away your password to a machine you don't trust anyway. Someone on that machine could just as easily change `login` to save all the passwords it sees into a file. This applies to `keylogin` too.

Not having `su` to employ any more, how can nefarious users impersonate others now? Probably the easiest way is to guess somebody's password, since most people don't choose very secure passwords. No ready-made protection can be offered against this; it's up to each user to choose a secure password.

The next best attack would be to attempt replays. For example, let's say user A has been squirreling away all of user B's NFS transactions with a particular server. As long as the server remains up, user A won't succeed by replaying them since the server always demands timestamps that are greater than the previous ones seen. But suppose user A goes and pull the plug on user B's server, causing it to crash. As it reboots, its credential table will be clean, so it has lost all track of previously seen timestamps, and now user A is free to replay user B's transactions. There are few things to be said about this. First of all, servers should be kept in a secure place so that no one can go and pull the plug on them. But even if they are physically secure, servers occasionally crash without any help. Replaying transactions is not a very big security problem, but even so, there is protection against it. If a client specifies a window size that is smaller than the time it takes a server to reboot (5 to 10 minutes), the server will reject any replayed transactions because they will have expired.

To specify a window size other than the default, you have to modify the kernel with `adb`:

```

# adb -w /vmunix -
authdes_win?D
_authdes_win:
_authdes_win: 3600
?W0t300
_authdes_win: 0xe10 = 0x12c
authdes_win?D
_authdes_win:
_authdes_win: 300
$g
#

```

In the above example, the superuser has changed the default window size from 3600 seconds (one hour) to 300 seconds (five minutes). Remember that if you were to reconfigure the kernel you would have to use `adb` again.

There are other ways to break DES authentication, but they are much more difficult. These methods involve breaking the DES key itself, or computing the logarithm of the public key. Generally, these methods are unfeasible for most crackers, although it is important to realize that any lock can be picked with a big enough hammer. What secure NFS provides is as secure as any good time-sharing system.

There is another security issue that DES authentication does not address, and that is tapping of the net. Even with DES authentication in place, there is no protection against somebody watching what goes across the net. There is no way in software to prevent unauthorized eavesdropping.

Performance

Public key systems are known to be slow, but there is not much actual public key encryption going on in this system. Public key encryption only occurs during the first transaction with a service, and even then, there is caching that speeds things up considerably. The first time a client program contacts a server, both it and the server will have to calculate the common key. The time it takes to compute the common key is basically the time it takes to compute an exponential modulo M . On a Sun-3 using a 192-bit modulus, this takes roughly 1 second, which means it takes 2 seconds just to get things started, since both client and server have to perform this operation. This is a long time, but you have to wait only the first time you contact a machine. Since the keyserver caches the results of previous computations, it does not have to recompute the exponential every time.

The most important service in terms of performance is secure NFS, which is acceptably fast. The extra overhead that DES authentication requires versus UNIX authentication is the encryption. A timestamp is a 64-bit quantity, which also happens to be the DES block size. Four encryption operations take place in an average RPC transaction: the client encrypts the request timestamp, the server decrypts it, the server encrypts the reply timestamp, and the client decrypts it. On a Sun-3, the time it takes to encrypt one block is about half a millisecond if performed by hardware, and 1.2 milliseconds if performed by software. So, the extra time added to the round trip time is about 2 milliseconds for hardware encryption and 5 for software. The round trip time for the average NFS request is

about 20 milliseconds, resulting in a performance hit of 10 percent if one has encryption hardware, and 25 percent if not. Remember that is the impact on network performance. The fact is that not all file operations go over the wire, so the impact on total system performance will actually be lower than this. It is also important to remember that security is optional, so environments that require higher performance can turn it off.

Problems with Booting and `setuid` Programs

Consider the problem of a machine rebooting, say after a power failure at some strange hour when nobody is around. All of the secret keys that were stored get wiped out, and now no process is able to access secure network services, such as mounting an NFS file hierarchy. The important processes at this time are usually root processes, so things would work OK if root's secret key were stored away, but nobody is around to type the password that decrypts it. The solution to this problem is to store root's decrypted secret key in a file, which the keyserver can read. This works well for diskful machines that can store the secret key on a physically secure local disk, but not so well for diskless machines, whose secret key must be stored across the network. If someone taps the net when a diskless machine is booting, they will find the decrypted key. This is not very easy to accomplish, though.

Another booting problem is the single-user boot. There is a mode of booting known as single-user mode, where a `root` login shell appears on the console. The problem here is that a password is not required for this. With C2 security installed, a password is required in order to boot single-user. Without C2 security installed, machines can still be booted single-user without a password, as long as the entry for `console` in the `/etc/ttytab` file is labeled as `physically secure` (this is the default). If the console is not physically secure, and you have security considerations, you should change the entry (and make sure you remember root's password). For more details, see the chapters *Administering Security* and *Administering C2 Security*.

Yet another problem is that diskless machine booting is not totally secure. It is possible for somebody to impersonate the boot-server, and boot a devious kernel that, for example, makes a record of a user's secret key on a remote machine. The problem is that the secure RPC system is set up to provide protection only after the kernel and the keyserver are running. Before that, there is no way to authenticate the replies given by the boot server. This is not considered a serious problem, because it is highly unlikely that somebody would be able to write this compromised kernel without source code. Also, the crime is not without evidence. If you polled the net for boot-servers, you would discover the devious boot-server's location. In any case, if you do not trust the network, you should boot from a local disk.

Not all `setuid` programs will behave as they should. For example, if a `setuid` program is owned by `dave`, who has not logged into the machine since it booted, then the program will not be able to access any secure network services as `dave`. The good news is that most `setuid` programs are owned by `root`, and since root's secret key is always stored at boot time, these programs will behave as they always have.

Conclusion

DES authentication satisfies the requirement of a networking environment as secure as a time-shared system. The way a user is authenticated in a time-sharing system is by knowing the password. With DES authentication, the same is true. In time-sharing the trusted person is the system administrator, who has an ethical obligation not to change a password in order to impersonate someone. In secure RPC, the network administrator is trusted not to alter entries in the public key database. In one sense, this system is even more secure than time-sharing, because it is useless to place a tap on the network in hopes of catching a password or encryption key, since these are encrypted. Most time-sharing environments do not encrypt data emanating from the terminal; users must trust that nobody is tapping their terminal lines.

DES authentication is perhaps not the ultimate authentication system. But at least DES authentication offers a smooth migration path for the future. Syntactically speaking, nothing in the protocol requires the encryption of the conversation key to be Diffie-Hellman, or even public key encryption in general. To make the authentication stronger in the future, all that needs to be done is to strengthen the way the conversation key is encrypted. Semantically, this will be a different protocol, but the beauty of RPC is that it can be plugged in and live peacefully with any authentication system.

For the present at least, DES authentication satisfies our requirements for a secure networking environment. From it we built a system secure enough for use in unfriendly networks, such as a student-run university workstation environment. The price for this security is not high. Nobody has to carry around a magnetic card or remember any hundred digit numbers. You use your login password to authenticate yourself, just as before. There is a small impact on performance, but if this worries you and you have a friendly net, you can turn authentication off.

14.10. Secure NFS References

Diffie and Hellman, "New Directions in Cryptography," *IEEE Transactions on Information Theory* IT-22, November 1976.

Gusella & Zatti, "TEMPO: A Network Time Controller for a Distributed Berkeley UNIX System," *USENIX 1984 Summer Conference Proceedings*, June 1984.

National Bureau of Standards, "Data Encryption Standard," *Federal Information Processing Standards Publication 46*, January 15, 1977.

Needham & Schroeder, "Using Encryption for Authentication in Large Networks of Computers," *Xerox Corporation CSL-78-4*, September 1978.

Using the NFS Automounter

15.1. The Automounter

The chapter *The Sun Network File System Service* explains how to export and mount file hierarchies through NFS with the `mount` command. You can also NFS mount file hierarchies using a different method—automounting. The `automount` program enables users to mount and unmount remote directories on an as-needed basis. Whenever a user on a client machine running the automounter invokes a command that needs to access a remote file or directory, such as opening a file with an editor, the hierarchy to which that file or directory belongs is mounted and remains mounted for as long as it is needed. And whenever a certain amount of time has elapsed without the hierarchy being accessed, it is automatically unmounted. No mounting is done at boot-time, and the user no longer has to know the superuser password to mount a directory or even use the `mount` and `umount` commands. It is all done automatically and transparently.

Mounting some file hierarchies with `automount` does not exclude the possibility of mounting others with `mount`. A diskless machine *must* mount `/`, `/usr` and `/usr/kvm` through the `mount` command and the `/etc/fstab` file. In all cases, the automounter should not be used to mount `/usr/share`.

This chapter explains how the automounter works, how to write the files (maps) the automounter uses, how to invoke it, and what are the error messages related to it. This chapter assumes that you have read *The Sun Network File System Service*.

How the Automounter Works

Unlike `mount`, `automount` does not consult the file `/etc/fstab` for a list of hierarchies to mount. Rather, it consults a series of maps, which can be either direct or indirect. The names of the maps can be passed to `automount` from the command line, or from another (master) map. This is explained later, in the section *Preparing the Maps*.

The following is a simplified bird's eye view of how the automounter works:

When `automount` is started, either from the command line or from `rc.local`, it forks a daemon to serve the mount points specified in the maps and makes the kernel believe that the mount has taken place. The daemon sleeps until a request is made to access the corresponding file hierarchy. At that time the daemon does the following:

- Intercepts the request

- Mounts the remote file hierarchy
- Creates a symbolic link between the requested mount point and the actual mount point under `/tmp_mnt`.
- Passes the symbolic link to the kernel, and steps aside.
- Unmounts the file hierarchy when a predetermined amount of time has passed with the link not being touched (generally five minutes), and resumes its previous position.

The automounter mounts everything under the directory `/tmp_mnt`, and provides a symbolic link from the requested mount point to the actual mount point under `/tmp_mnt`. For instance, if a user wants to mount a remote directory `src` under `/usr/src`, the actual mount point will be `/tmp_mnt/usr/src`, and `/usr/src` will be a symbolic link to that location. Note that, as with any other kind of mount, a mount affected through the automounter on a non-empty mount point will hide the original contents of the mount point for as long as the mount is in effect.

The `/tmp_mnt` directory is created automatically by the automounter. Its default name can be changed, as explained later in this chapter.

15.2. Preparing the Maps

A server never knows, nor cares, whether the files it exports are accessed through mount or automount. Therefore, you need not do anything different on the server for automount than for mount.

A client, however, needs special files for the automounter. As mentioned previously, automount does not consult `/etc/fstab`; rather, it consults the map file(s) specified on the command line (see below, *Starting automount*). If no maps are specified, it looks for an NIS map called `auto.master` (The chapter *The Network Information Service* explains how to make this map). This is, in fact, how it is invoked by default, through `/etc/rc.local`:

```
if [ -f /usr/etc/automount ]; then
    automount && echo -n ' automount'
fi
```

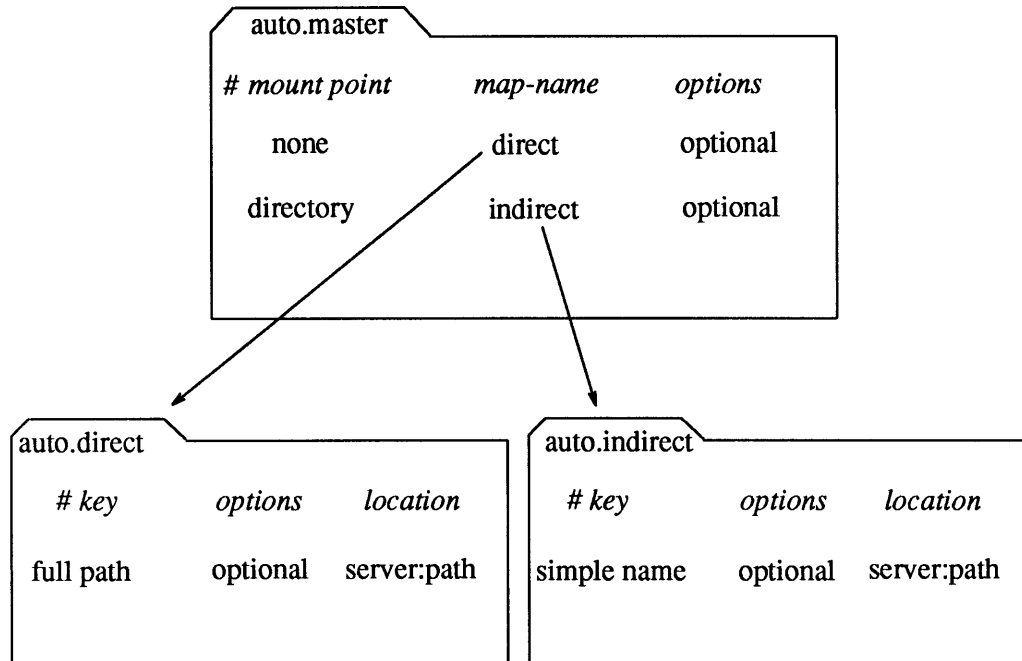
If no NIS `auto.master` exists, automount exits silently. Later on, in this chapter, we explain how to invoke automount so that it consults local maps.

By convention, all automounter maps are located in the directory `/etc` and their names all begin with the prefix `auto`.

There are three kinds of automount maps:

- master
- indirect
- direct

The *master* map lists (as if from the command line) all other maps, applicable options and mount points, as described below and summarized in the following figure:

Figure 15-1 *Master, Direct, and Indirect Maps*

The Master Map

Each line in a master map, by convention called `/etc/auto.master`, has the syntax:

```
mount-point    map-name    [ mount-options ]
```

where:

- *mount-point* is the full pathname of a directory. If the directory does not exist, the automounter will create it if possible. If the directory exists, and is not empty, mounting on it will hide its contents. The automounter will issue a warning message in this case.
- *map-name* is the map the automounter should use to find the mount points and locations.
- *mount-options* is an optional, comma separated, list of options that regulate the mounting of the entries mentioned in *map-name*, unless the entries in *map-name* list other options.

A line whose first character is `#` is treated as a comment, and everything that follows until the end of line is ignored. A backslash (`\`) at the end of line permits splitting long lines into shorter ones. The notation `/-` as a mount point indicates that the map in question is a direct map, and no particular mount point is associated with the map as a whole.

Direct and Indirect Maps

Lines in direct and indirect maps have the syntax:

```
key [ mount-options ] location
```

where

- *key* is the pathname of the mount point.
- The *mount-options* are the options you want to apply to this particular mount.
- *location* is the location of the resource, specified as *server:pathname*.

As in the master map, a line whose first character is # is treated as a comment and everything that follows until the end of line is ignored. A backslash at the end of line permits splitting long lines into shorter ones.

The only formal difference between a direct and an indirect map is that the key in a direct map is a full pathname, whereas in an indirect pathname it is a simple name (no slashes). For instance, the following would be an entry in a direct map:

```
/usr/man -ro,intr goofy:/usr/man
```

and the following would be an entry in an indirect map:

```
parsley -ro,intr veggies:/usr/greens
```

As you can see, the *key* in the indirect map is begging for more information: where is the mount point *parsley* really located? That is why you must either provide that information at the command line or through another map. For instance, if the above line is part of a map called */etc/auto.veggies*, you would have to invoke it either as:

```
automount /veggies /etc/auto.veggies
```

or specify, in the master map:

```
/veggies /etc/auto.veggies -ro,soft,nosuid
```

In either case, you are associating a mount directory (*/veggies*) with the entries (*parsley* in this case) mentioned in the indirect map */etc/auto.veggies*. The end result is that the hierarchy */usr/greens* from the machine *veggies* will be mounted on */veggies/parsley* when needed.

Writing a Master Map

As stated above, the syntax for each line in the master map is

```
mount-point      map-name          [ mount-options ]
```

A typical `auto.master` file would contain

```
#Mount-point  Map                Mount-options
/-            /etc/auto.direct  -ro,intr
/home        /etc/auto.home    -rw,intr,secure
/net         -hosts
```

Figure 15-2 *Sample auto.master Map*

The automounter recognizes some special mount points and maps, which are explained below.

Mount point /-

In the example above, the mount point `/-` is a filler that the automounter recognizes as a directive not to associate the entries in `/etc/auto.direct` with any directory. Rather, the mount points are to be the ones mentioned in the map. (Remember, in a direct map the key is a full pathname.)

Mount point /home

The mount point `/home` is to be the directory under which the entries listed in `/etc/auto.home` (an indirect map) are to be mounted. That is, they will be mounted under `/tmp_mnt/home`, and a symbolic link will be provided between `/home/directory` and `/tmp_mnt/home/directory`.

Mount point /net

Finally, the automounter will mount under the directory `/net` all the entries under the special map `-hosts`. This is a built-in map that does not use any external files except the hosts database (`/etc/hosts` or the NIS map `hosts.byname` if NIS is running) Notice that since the automounter does not mount the entries until needed, the specific order is not important. Once the automount daemon is in place, a user entering the command

```
example % cd /net/gumbo
```

will change directory to the top of the hierarchy of files (i.e., the root file system) of the machine `gumbo` as long as the machine is in the hosts database and it exports any of its file systems. However, the user may not see under `/net/gumbo` all the files and directories. This is because the automounter can mount only the *exported* file systems of host `gumbo`, in accordance with the restrictions placed on the exporting.

The actions of the automounter when the command in the example above is issued are as follows:

1. `ping` the null procedure of the server's mount service to see if it's alive.

2. Request the list of exported hierarchies from the server.
3. Sort the exported list according to length of pathname.

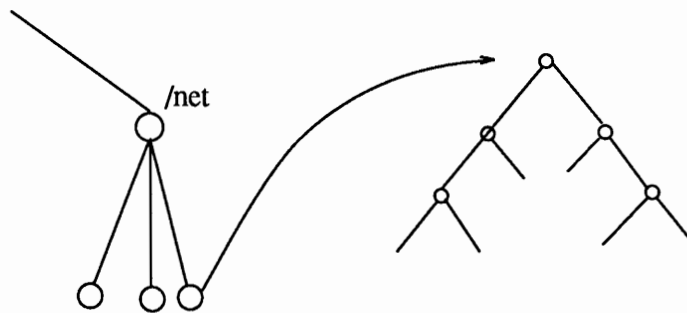
```

/usr/src
/export/home
/usr/src/sccs
/export/root/blah

```

This sorting ensures that the mounting is done in the proper order, that is, `/usr/src` is done before `/usr/src/sccs`.

4. Proceed down the list, mounting all the file systems at mount points in `/tmp_mnt` (creating the mount points as needed).
5. Return a symbolic link that points to the top of the recently mounted hierarchy:



Note that, unfortunately, the automounter has to mount all the file systems that the server in question exports. Even if the request is as follows:

```
example % ls /net/gumbo/usr/include
```

the automounter mounts all of `gumbo`'s exported systems, not just `/usr`.

In addition, unmounting that occurs after a certain amount of time has passed is from the bottom up. This means if one of the directories at the top is busy, the automounter has to remount the hierarchy and try again later.

Nevertheless, the `-hosts` special map provides a very convenient way for users to access directories in many different hosts without having to use `rlogin` or `rsh`. (These remote commands have to establish communication through the network every time they are invoked.) Furthermore, they no longer have to modify their `/etc/fstab` files or mount the directories by hand as superuser.

Notice that both `/net` and `/home` are arbitrary names dictated by convention. The automounter will create them if they do not exist already.

Writing an Indirect Map

The syntax for an indirect map is:

```
key      [ mount-options ]      location
```

where *key* is the basename (not the full pathname) of the directory that will be used as mount point. Once the key is obtained by the automounter, it is suffixed to the mount point associated with it either by the command line or by the master map that invokes the indirect map in question.

For instance, one of the entries in the master map presented above as an example, reads:

```
/home      /etc/auto.home  -rw,intr,secure
```

Here `/etc/auto.home` is the name of the indirect map that will contain the entries to be mounted under `/home`.

A typical `auto.home` map might contain:

```
#key      mount-options  location
willow
cypress
poplar
pine
apple
ivy
peach     -rw,nosuid  peach:/export/home
willow:/home/willow
cypress:/home/cypress
poplar:/home/poplar
pine:/export/pine
apple:/export/home
ivy:/home/ivy
```

Figure 15-3 *Sample auto.home Map*

As an example, assume that the map above is on host *oak*. If user *laura* has an entry in the password database specifying her home directory as `/home/willow/laura`, then whenever she logs into machine *oak*, the automounter will mount (as `/tmp_mnt/home/willow`) the directory `/home/willow` residing in machine *willow*. If one of the directories is indeed *laura*, she will be in her home directory, which is mounted read/write, interruptible and secure, as specified by the options field in the master map entry.

Suppose, however, that *laura*'s home directory is specified as `/home/peach/laura`. Whenever she logs into *oak*, the automounter mounts the directory `/export/home` from *peach* under `/tmp_mnt/home/peach`. Her home directory will be mounted read/write, `nosuid`. *Any option in the file entry overrides all options in the master map or the command line.*

Now, assume the following conditions occur:

- User *laura*'s home directory is listed in the password database as `/home/willow/laura`.

- Machine willow exports its home hierarchy to the machines mentioned in `auto.home`.
- All those machines have a copy of the same `auto.home` and the same password database.

Under these conditions, user `laura` can run `login` or `rlogin` on any of these machines and have her home directory mounted in place for her.

Furthermore, now `laura` can also enter the command

```
% cd ~brent
```

and the automounter will mount `brent`'s home directory for her (if all permissions apply).

On a network without NIS, you have to change all the relevant databases (such as `/etc/passwd`) on all systems on the network in order to accomplish this. On a network running NIS, make the changes on the NIS master server and propagate the relevant databases to the slave servers.

Writing a Direct Map

The syntax for a direct map (like that for an indirect map) is:

```
key [ mount-options ] location
```

where:

- *key* is the *full* pathname of the mount point. (Remember that in an indirect map this is not a full pathname.)
- *mount-options* are optional but, if present, override — for the entry in question — the options of the calling line, if any, or the defaults. (See below, *Starting automount*).
- *location* is the location of the resource, specified as *server:pathname*.

Of all the maps, the entries in a direct map most closely resemble, in their simplest form, what their corresponding entries in `/etc/fstab` might look like. An entry that appears in `/etc/fstab` as:

```
dancer:/usr/local /usr/local/tmp nfs ro 0 0
```

appears in a direct map as:

```
/usr/local/tmp -ro dancer:/usr/local
```

The following is a typical `/etc/auto.direct` map:

```

/usr/local \
        /bin      -ro,soft  ivy:/export/local/sun3 \
        /share   -ro,soft  ivy:/export/local/share \
        /src     -ro,soft  ivy:/export/local/src
/usr/man
        -ro,soft  oak:/usr/man \
        rose:/usr/man \
        willow:/usr/man
/usr/games
        -ro,soft  peach:/usr/games
/usr/spool/news
        -ro,soft  pine:/usr/spool/news
/usr/frame
        -ro,soft  redwood:/usr/frame2.0 \
        balsa:/export/frame

```

Figure 15-4 *Sample auto.direct Map*

There are a couple of unusual features in this map: multiple mounts and multiple locations. These are the subject of the next two subsections.

Multiple Mounts

A map entry can describe a multiplicity of mounts, where the mounts can be from different locations and with different mount options. Consider the first entry in the previous example:

```

/usr/local \
        /bin      -ro,soft  ivy:/export/local/sun3 \
        /share   -ro,soft  ivy:/export/local/share \
        /src     -ro,soft  ivy:/export/local/src

```

This is, in fact, one long entry whose readability has been improved by splitting it into four lines by using the backslash and indenting the continuation lines with blank spaces or tabs. This entry mounts `/usr/local/bin`, `/usr/local/share` and `/usr/local/src` from the server `ivy`, with the options read-only and soft. The entry could also read:

```

/usr/local \
        /bin      -ro,soft  ivy:/export/local/sun3 \
        /share   -rw,secure willow:/usr/local/share \
        /src     -ro,intr  oak:/home/jones/src

```

Figure 15-5 *Example of a Multiple Mount*

where the options are different and more than one server is used. The difference between the above and three separate entries, for example:

```

/usr/local/bin      -ro,soft    ivy:/export/local/sun3
/usr/local/share   -rw,secure  willow:/usr/local/share
/usr/local/src     -ro,intr   oak:/home/jones/src

```

is that the first case, multiple mount, guarantees that all three directories will be mounted when you reference one of them. In the case of the separate entries, if you, for instance, enter:

```
% cd /usr/local/bin
```

you cannot `cd` to one of the other directories using a relative path, because it is not mounted yet:

```
% cd ../src
../src: No such file or directory
```

A multiple mount obviates this problem. In multiple mounts, each file hierarchy is mounted on a subdirectory within another file hierarchy. When the root of the hierarchy is referenced, the automounter mounts the whole hierarchy.

The concept of *root* here is very important. The symbolic link returned by the automounter to the kernel request is a path to the mount root. This is the root of the hierarchy that is mounted under `/tmp_mnt`. This mount point should theoretically be specified:

```
parsley / -ro,intr veg:/usr/greens
```

In practice, it is not specified because in a trivial case of a single mount as above, it is assumed that the location of the mount point is *at* the mount root or `“/.”` So instead of the above it is perfectly acceptable, indeed preferable, to enter:

```
parsley -ro,intr veg:/usr/greens
```

The mount point specification, however, becomes important when mounting a hierarchy: here the automounter must have a mount point for each mount within the hierarchy. The example above is a good illustration of multiple, non-hierarchical mounts under `/usr/local` when the latter is already mounted (or is a subdirectory of another mounted system).

When the root of the hierarchy has to be mounted also, it has to be specified in the map, and the entry becomes a “hierarchical” mount, which is a special case of multiple mounts.

The following illustration shows a true hierarchical mounting:


```

/usr/local \
/           -rw,intr    peach:/export/local \
/bin       -ro,soft    ivy:/export/local/sun3 \
/share     -rw,secure  willow:/usr/local/share \
/src       -ro,intr    oak:/home/jones/src

```

Figure 15-6 *Example of a Hierarchical Multiple Mount*

Note: A true hierarchical mount can be problematic if the server for the root of the hierarchy goes down. Any attempt to unmount the lower branches will fail, since the unmounting has to proceed through the mount root, which also cannot be unmounted while its server is down.

The mount points used here for the hierarchy are `/`, `/bin`, `/share`, and `/src`. Note that these mount point paths are relative to the *mount* root, not the host's *file system* root. The first entry in the example above has `/` as its mount point. It is mounted *at* the mount root. There is no requirement that the first mount of a hierarchy be at the mount root. The automounter will happily issue `mkdir`'s to build a path to the first mount point if it is not at the mount root.

Finally, a word about mount options. In one of the examples above,

```

/usr/local \
/bin       -ro,soft    ivy:/export/local/sun3 \
/share     -ro,soft    willow:/usr/local/share \
/src       -ro,soft    oak:/home/jones/src

```

all three mounts share the same options. This could be modified to:

```

/usr/local  -ro,soft \
/bin       ivy:/export/local/sun3 \
/share     willow:/usr/local/share \
/src       oak:/home/jones/src

```

If one of the mount points needed a different specification, you could then write:

```

/usr/local  -ro,soft \
/bin       ivy:/export/local/sun3 \
/share     -rw,secure  willow:/usr/local/share \
/src       oak:/home/jones/src

```

Multiple Locations

In the example for a direct map, which was:

```

/usr/local \
                /bin      -ro,soft  ivy:/export/local/sun3 \
                /share    -ro,soft  ivy:/export/local/share \
                /src      -ro,soft  ivy:/export/local/src
/usr/man      -ro,soft  oak:/usr/man \
                rose:/usr/man \
                willow:/usr/man
/usr/games      -ro,soft  peach:/usr/games
/usr/spool/news -ro,soft  pine:/usr/spool/news
/usr/frame    -ro,soft  redwood:/usr/frame2.0 \
                balsa:/export/frame

```

the mount points `/usr/man` and `/usr/frame` (bolded in the example above) list more than one location (three for the first, two for the second). This means that the mounting can be done from any of the replicated locations. This procedure makes sense only when you are mounting a hierarchy read-only, since (at least theoretically) you must have some control over the locations of files you write or modify (that is, you don't want to modify files on one server on one occasion and, minutes later, modify the "same" file on another server). A good example is the man pages. In a large network, more than one server may export the current set of manual pages. It does not matter which server you mount them from, as long as the server is up and running and exporting its file systems. In the example above, multiple mount locations are expressed as a list of mount locations in the map entry:

```
/usr/man -ro,soft oak:/usr/man rose:/usr/man willow:/usr/man
```

This could also be expressed as a comma separated list of servers, followed by a colon and the pathname (as long as the pathname is the same for all the replicated servers):

```
/usr/man -ro,soft oak,rose,willow:/usr/man
```

Here you can mount the man pages from the servers *oak*, *rose*, or *willow*. From this list of servers the automounter first selects those that are on the local network and pings these servers. This launches a series of RPC requests to each server. The first server to respond is selected, and an attempt is made to mount from it. Note that the list does not imply an ordering.

This redundancy, very useful in an environment where individual servers may or may not be exporting their file systems, is enjoyed only at mount time. There is no status checking of the mounted-from server by the automounter once the mount occurs. If the server goes down while the mount is in effect, the file system becomes unavailable. An option here is to wait five minutes until the

auto-unmount takes place and try again. Next time around the automounter will choose one of the other, available servers. Another option is to use the `umount` command, inform the automounter of the change in the mount table (as specified in the later section, *The Mount Table*, and retry the mount.

Specifying Subdirectories

The earlier subsection, *Writing an Indirect Map*, showed the following typical `auto.home` file:

```
#key      mount-options  location
willow    willow:/home/willow
cypress   cypress:/home/cypress
poplar    poplar:/home/poplar
pine      pine:/export/pine
apple     apple:/export/home
ivy       ivy:/home/ivy
peach     -rw,nosuid    peach:/export/home
```

Given this `auto.home` indirect file, every time a user wants to access a home directory in, say, `/home/willow`, all the directories under it will be mounted. Another way to organize an `auto.home` file is by user name, as in:

```
#key      mount-options  location
john      willow:/home/willow/john
mary      willow:/home/willow/mary
joe       willow:/home/willow/joe
```

The above example assumes that home directories are of the form `/home/user` rather than `/home/server/user`. If a user now enters the following command:

```
% ls ~john ~mary
```

the automounter has to perform the *equivalent* of the following actions:

```
mkdir /tmp_mnt/home/john
mount willow:/home/willow/john /tmp_mnt/home/john
ln -s /tmp_mnt/home/john /home/john

mkdir /tmp_mnt/home/mary
mount willow:/home/willow/mary /tmp_mnt/home/mary
ln -s /tmp_mnt/home/mary /home/mary
```

However, the complete syntax of a line in a direct or indirect map is actually:

```
key [ mount-option ] server:pathname[:subdirectory]
```

Until now you used the form *server:pathname* to indicate the location. This is an ideal place for you to also indicate the subdirectory, like this:

```
#key  mount-options  location
john  willow:/home/willow:john
mary  willow:/home/willow:mary
joe   willow:/home/willow:joe
```

Figure 15-7 *Example of subdirectory Field*

Here *john*, *mary* and *joe* are entries in the *subdirectory* field. Now when a user refers to *john's* home directory, the automounter mounts *willow:/home/willow*. It then places a symbolic link between */tmp_mnt/home/willow/john* and */home/john*.

If the user then requests access to *mary's* home directory, the automounter sees that *willow:/home/willow* is already mounted, so all it has to do is return the link between */tmp_mnt/home/willow/mary* and */home/mary*. In other words, the automounter now only does:

```
mkdir /tmp_mnt/home/john
mount willow:/home/willow /tmp_mnt/home
ln -s /tmp_mnt/home/john /home/john

ln -s /tmp_mnt/home/mary /home/mary
```

In general, it is a good idea to provide a *subdirectory* entry in the *location* when different map entries refer to the same mounted file system from the same server.

Metacharacters

The automounter recognizes some characters as having a special meaning. Some are used for substitutions, some to escape other characters.

Ampersand (&)

If you have a map with a lot of subdirectories specified, like:

```
#key  mount-options  location
john  willow:/home/willow:john
mary  willow:/home/willow:mary
joe   willow:/home/willow:joe
able  pine:/export/home:able
baker peach:/export/home:baker

[. . .]
```

consider using string substitutions. You can use the ampersand character (&) to substitute the key wherever it appears. Using the ampersand, the above map now looks as follows:

Figure 15-8 *Example of Use of Ampersand*

```

#key      mount-options  location
john      willow:/home/willow:&
mary      willow:/home/willow:&
joe       willow:/home/willow:&
able      pine:/export/home:&
baker     peach:/export/home:&
          [. . .]

```

If the name of the server is the same as the key itself, for instance:

```

#key      mount-options  location
willow    willow:/home/willow
peach     peach:/home/peach
pine      pine:/home/pine
oak       oak:/home/oak
poplar    poplar:/home/poplar
          [. . .]

```

the use of the ampersand results in:

```

#key      mount-options  location
willow    &:/home/&
peach     &:/home/&
pine      &:/home/&
oak       &:/home/&
poplar    &:/home/&
          [. . .]

```

Asterisk (*)

Notice that all the above entries have the same format. This permits you to use the catch-all substitute character, the asterisk (*). The asterisk reduces the whole thing to:

Figure 15-9 *Example of Use of Ampersand and Asterisk*

```

*      &:/home/&

```

where each ampersand is substituted by the value of any given key. Notice that once the automounter reads the catch-all key it does not continue reading the map, so that the following map would be viable:

```
#key      mount-options  location
oak                               &:/export/&
poplar   &:/export/&
*        &:/home/&
```

whereas in the next map the last two entries would always be ignored:

```
#key      mount-options  location
*        &:/home/&
oak       &:/export/&
poplar    &:/export/&
```

You could also use key substitutions in a direct map, in situations like the following:

```
/usr/man      willow,cedar,poplar:/usr/man
```

which is a good candidate to be written as:

```
/usr/man      willow,cedar,poplar:&
```

Notice that the ampersand substitution uses the whole key string, so if the key in a direct map starts with a / (as it should), that slash is carried over, and you could not do something like

```
/progs  &1,&2,&3:/export/src/progs
```

because the automounter would interpret it as:

```
/progs  /progs1,/progs2,/progs3:/export/src/progs
```

Backslash (\)

Under certain circumstances you may have to mount directories whose names may confuse the automounter's map parser. An example might be a directory called `rc0:dk1`; this could result in an entry like:

```
/junk      -ro      vmserver:rc0:dk1
```

The presence of the two colons in the location field will confuse the automounter's parser. To avoid this confusion, use a backslash to escape the second colon and remove its special meaning of separator:

```
/junk -ro vmserver:rc0\ :dk1
```

Double quotes (")

You can also use double quotes, as in the following example, where they are used to hide the blank space in the name:

```
/smile dentist:/"front teeth"/smile
```

Environment Variables

You can use the value of an environmental variable by prefixing a dollar sign (\$) to its name. You can also use braces to delimit the name of the variable from appended letters or digits. You can use environmental variables anywhere in an entry line, except as a *key*. An interesting variable is \$ARCH, whose value is the name of the machine's architecture as would be returned by the command `arch`. For instance, if you have a file server exporting binaries for Sun-3 and Sun-4 architectures from `/usr/local/bin/sun3` and `/usr/local/bin/sun4` respectively, you can have the clients mount through a map entry like the following:

```
/usr/local/bin -ro server:/usr/local/bin/$ARCH
```

Now the same entry on all the clients is good for all architectures.

The environmental variables can be inherited from the environment or can be defined explicitly with the `-D` command line option. For instance, if you want each client to mount client-specific files in the network in a replicated format, you could create a specific map for each client according to its name, so that the relevant line for host oak would be:

```
/mystuff cypress,ivy,balsa:/export/hostfiles/oak
```

and for willow it would be:

```
/mystuff cypress,ivy,balsa:/export/hostfiles/willow
```

This scheme is viable within a small network, but maintaining this kind of host specific maps across a large network would soon become unfeasible. The solution in this case would be to start the automounter with a command line similar to the following:

```
automount -D HOST='hostname' .....
```

and have the entry in the direct map read:

```
/mystuff cypress,ivy,balsa:/export/hostfiles/$HOST
```

Now each host would find its own files in the `mystuff` directory, and the task of centrally administering and distributing the maps becomes easier.

Similarly, if you want to mount using `arch -k` as the criterion, you could start the automounter as:

```
automount -D ARCH='arch -k' .....
```

and still use the entry

```
/usr/local/bin -ro server:/usr/local/bin/$ARCH
```

(or a similar one).

Including Other Maps

A line of the form `+mapname` causes the automounter to consult the mentioned map as if it were included in the current map. If `mapname` is a relative pathname (no slashes), the automounter assumes it is an NIS map. If the pathname is an absolute pathname the automounter looks for a local map of that name. If the mapname starts with a dash (`-`), the automounter consults the appropriate built-in map.

For instance, you can have a few entries in your local `auto.home` map for the most commonly accessed home directories, and follow them with the included NIS map:

```
ivy      -rw,intr,noquota  &:/home/&
oak      -rw,intr,noquota  &:/export/home
+auto.home
```

After consulting the included map, the automounter continues scanning the current map if no match is found, so you can add more entries, for instance:

```
ivy      -rw,intr,noquota  &:/home/&
oak      -rw,intr,noquota  &:/export/home
+auto.home
*        -rw              &:/home/&
```

Finally, as mentioned before, the map included can be a local file, or even a built-in map:

```
+auto.home.finance      # NIS map
+auto.home.sales        # NIS map
+auto.home.engineering  # NIS map
+/etc/auto.mystuff      # local map
+auto.home              # NIS map
+-hosts                 # built-in hosts map
* &:/export/&           # wild card
```

Notice that in all cases the wild card should be the last entry, because the automounter does not continue consulting the map after it, on the assumption that the

wild card will have found a match.

15.3. Starting automount

Once the maps are written, you should make sure that there are no equivalent entries in `/etc/fstab`, and that all the entries in the maps refer to NFS exported files.

The syntax to invoke the automounter is:

```
automount [ -mTv ] [ -D name = value ] [ -f master-file ] [ -M mount-directory ]
[ -tl duration ] [ -tm interval ] [ -tw interval ] [ directory map [ -mount-options ] ] ...
```

The `automount(8)` man page contains a complete description of all options.

The *mount-options* that you can specify at either the command line or in the maps are the same as those for a standard NFS mount, excepting `bg` (background) and `fg` (foreground), which do not apply.

By default, the file `/etc/rc.local` starts the automounter at boot time through the lines:

```
if [ -f /usr/etc/automount ]; then
    automount && echo -n ' automount'
fi
```

That is, if the file `/usr/etc/automount` exists, start it. When started like this, with no option, the automounter looks for an NIS map called `auto.master`. If it finds it, it follows the instructions contained there. If NIS is not running, or the map is not to be found, the automounter exits silently. The `-m` option instructs the automounter not to look for the NIS map. The `-f` option instructs it to look for the file named immediately after the option. If no `-m` or `-f` options are specified, the automounter expects a series of mount points and maps (and optional mount options) specified on the command line.

Given the following set of three maps:

- `auto.master`

#Mount-point	Map	Mount-options
/net	-hosts	
/home	/etc/auto.home	-rw,intr,secure
/-	/etc/auto.direct	-ro,intr

□ auto.home

#key	mount-options	location
willow		willow:/home/willow
cypress		cypress:/home/cypress
poplar		poplar:/home/poplar
pine		pine:/export/pine
apple		apple:/export/home
ivy		ivy:/home/ivy
peach	-rw,nosuid	peach:/export/home

□ auto.direct

```

/usr/local \
                /bin      -ro,soft  ivy:/export/local/sun3 \
                /share   -ro,soft  ivy:/export/local/share \
                /src     -ro,soft  ivy:/export/local/src
/usr/man
                -ro,soft  oak:/usr/man \
                rose:/usr/man \
                willow:/usr/man
/usr/games
                -ro,soft  peach:/usr/games
/usr/spool/news
                -ro,soft  pine:/usr/spool/news
/usr/frame
                -ro,soft  redwood:/usr/frame1.3 \
                balsa:/export/frame

```

you can invoke the automounter (either from the command line or, preferably, from `rc.local`) in one of the following ways:

1. You can specify all arguments to the automounter without reference to the master map (either NIS or local), as in:

```
automount /net -hosts /home /etc/auto.home -rw,intr,secure /- /etc/auto.direct -ro,intr
```

2. You can specify the same in the `auto.master` file, and instruct the automounter to look in it for instructions:

```
automount -f /etc/auto.master
```

3. You can specify more mount points and maps in addition to those mentioned in the master map, as follows:

```
automount -f /etc/auto.master /src /etc/auto.src -ro,soft
```

OR

```
automount /src /etc/auto.src -ro,soft
```

The first example specifies additions to the local master map, the second additions to the NIS master map.

4. You can nullify one of the entries in the master map. (This is particularly useful if you use a map that you cannot modify and does not meet the needs of your machine):

```
automount -f /usr/lib/auto.master /home -null
```

This cancels the entry for /home in the master map (it could also be the NIS master map).

5. You can replace one of the entries with your own:

```
automount -f /usr/lib/auto.master /home /myown/auto.home -rw,intr
```

In the example above, the automounter first mounts all items in the map /myown/auto.home under the directory /home. Then, when it consults the master file /usr/lib/auto.master and reaches the line corresponding to /home it simply ignores it, since it has already mounted on it.

Given the auto.master file of the previous example, commands (1) and (2) are equivalent *as long as your network does not have a distributed auto.master file*. This file is only available on networks running NIS. If your network includes a distributed NIS auto.master file, the second example would have to be modified in the following way to be equivalent to example 1:

```
automount -m -f /etc/auto.master
```

The -m option instructs the automounter not to consult the master file distributed by NIS. However, if you do not run NIS, you do not have to specify the -m option. The automounter is completely silent when it does not find a distributed master file.

You can log in as superuser and type any of the above commands at shell level to start the automounter. Ideally, you should edit rc.local and include your preferred command there.

The Temporary Mount Point

The default name for all mounts is /tmp_mnt. Like the other names, this is arbitrary. It can be changed at startup time by use of the -M option. For instance:

```
automount -M /auto ...
```

causes all mounts to happen under the directory /auto, which the automounter will create if it does not exist. It goes without saying that you should not designate a directory in a read only file system, as the automounter would not be able

to modify anything then.

The Mount Table

Every time the automounter mounts or unmounts a file hierarchy, it modifies `/etc/mtab` to reflect the current situation. The automounter keeps an image in memory of `/etc/mtab`, and refreshes this image every time it performs a mounting or an automatic unmounting. If you use the `umount` command to unmount one of the automounted hierarchies (a directory under `/tmp_mnt`), the automounter should be forced to re-read the `/etc/mtab` file. To do that, enter the command:

```
example % ps -ef | grep automount | egrep -v grep
```

This gives you the process ID of the automounter. The automounter is designed so that on receiving a `SIGHUP` signal it re-reads `/etc/mtab`. So, to send it that signal, enter:

```
% kill -1 PID
```

where `PID` stands for the process ID you obtained from the previous `ps` command.

Modifying the Maps

You can modify the automounter maps at any time, but that does not guarantee that all your modifications will take effect the next time the automounter mounts a hierarchy. It depends on what map you modify and what kind of modification you introduce. You may have to reboot the machine. This is generally the simplest way of restarting the automounter, although, if it is used sparingly, you could theoretically kill it and restart it from the command line.

Modifying the Master Map

The automounter consults the master map only at startup time. A modification to the master map will take effect only next time you reboot the machine.

Modifying Indirect Maps

Entries can be modified, deleted, or added to indirect maps and the change will take effect next time the map is used, which is next time a mount has to be done.

Modifying Direct Maps

Each entry in a direct map is an automount mount point, and it only mounts itself at these mount points at startup. Therefore, adding or deleting an entry in a direct map will take effect only next time you reboot the machine. However, existing entries can be modified (changing mount options or server names, and so on, but not name of mount points) while the automounter is running, and will take effect when the entry is next mounted, because the automounter consults the direct maps whenever a mount has to be done.

For instance, suppose you modify the file `/etc/auto.direct` so that the directory `/usr/src` is now mounted from a different server. The new entry takes effect immediately (if `/usr/src` is not mounted at this time) when you try to access it. If it is mounted now, you can wait until the auto unmounting takes place, and then access it. If this is not satisfactory, you can unmount with the `umount` command, notify `automount` that the mount table has changed (see above, *The Mount Table*), and then access it. The mounting should now be

done from the new server. However, if you wanted to delete the entry, you would have to reboot the machine for the deletion to take effect.

For this reason, and because they do not clutter the mount table like direct maps do, indirect maps are preferable, and should be used whenever possible.

Mount Point Conflicts

If you have a home partition, on a local disk, that is mounted on `/home` and you want also to use the automounter to mount other home directories, you may encounter the problem that if you give it the mount point `/home` then the automounter will hide the local home partition whenever you try to reach it.

The solution is to mount the partition somewhere else, for instance on `/export/home`. You would then need, for instance, an entry in `/etc/fstab` that says:

```
/dev/xx#z ... /export/home ...
```

(where `xx#z` stands for the name of the partition) and, assuming that the master file contains a line similar to:

```
/home /etc/auto.home
```

an entry in `auto.home` that says:

```
terra terra:/export/home
```

where `terra` is the name of the machine.

If the partition is set up such as home directories are to be found as `/home/machine/user`, move all the directories at the `user` level one level up, so as to eliminate the `machine` level:

```
# cd /home
# mv machine/* .
# rmdir machine
```

There is no need to change the `/etc/passwd` entry for the user. His or her home directory will still be accessible through `/home/machine/user`, as before. Instead of doing a mount, the automounter will recognize that the file system is on the same machine and will establish a symbolic link from `/home/machine` to `/export/home`.

15.4. Problem Solving

From time to time you may encounter problems with the automounter. This section is aimed at making the problem solving process easier. It is divided in two subsections.

The first subsection expands on the bird's eye view explanation of how the automounter works presented at the beginning of this chapter. This explanation is written especially for advanced system administrators and programmers, and you can skip it without your ability to administer the automounter being affected. However, you may want to read it so as to have an idea of the issues involved.

The second subsection presents a list of the error messages the automounter generates. The list is divided in two parts:

- Error messages generated by the verbose (`-v`) option of `automount`, and
- Error messages that may appear at any time.

In general, it is recommended that you start the automounter with the verbose option, since otherwise you may experience problems without knowing why.

automount Sequence of Events

There are two distinct stages in the automounter's actions:

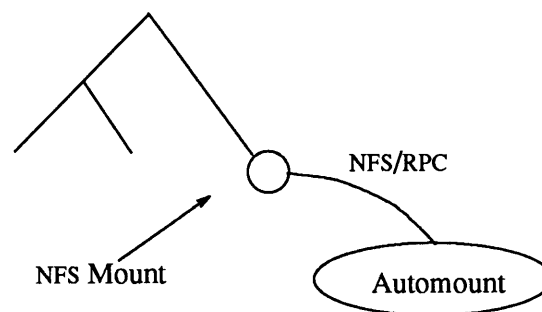
- The initial stage, boot time, when `rc.local` boots the automounter.
- The mounting stage, when a user tries to access a file or directory in a remote machine.

At the initial stage, when `rc.local` invokes `automount`, it opens a UDP socket and registers it with the `portmapper` service as an NFS server port. It then forks off a server daemon that listens for NFS requests on the socket. The parent process proceeds to mount the daemon at its mount points within the file system (as specified by the `maps`). Through the `mount(2)` system call it passes the server daemon's port number and an NFS *file handle* that is unique to each mount point. The arguments to the `mount(2)` system call vary according to the kind of file system; for NFS file systems, the call is

```
mount ( "nfs", "/usr", &nfs_args );
```

where `&nfs_args` contains the network address for the NFS server. By having the network address in `&nfs_args` refer to the local process (the `automount` daemon), `automount` in fact deceives the kernel into treating it as if it were an NFS server. Instead, once the parent process completes its calls to `mount(2)` it exits, leaving the daemon to serve its mount points:

Figure 15-10 Automount Disguised as NFS Server



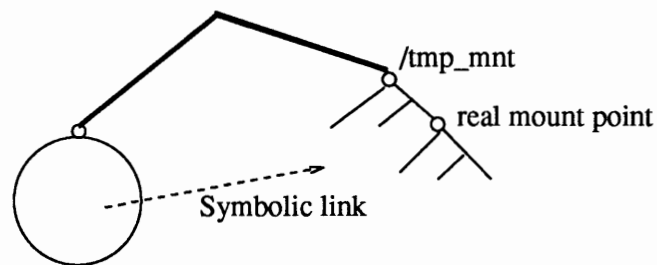
In the second stage, when the user actually requests access to a remote file hierarchy, the daemon intercepts the kernel NFS request and looks up the name in the map associated with the directory. Taking the location (*server:pathname*) of the remote file system from the map, the daemon then mounts the remote file system under the directory `/tmp_mnt`. It answers the kernel, telling it is a symbolic

link. The kernel sends an NFS READLINK request, and the automounter returns a symbolic link to the real mount point under `/tmp_mnt`.

The behavior of the automounter is affected by whether the name is found in a direct or an indirect map.

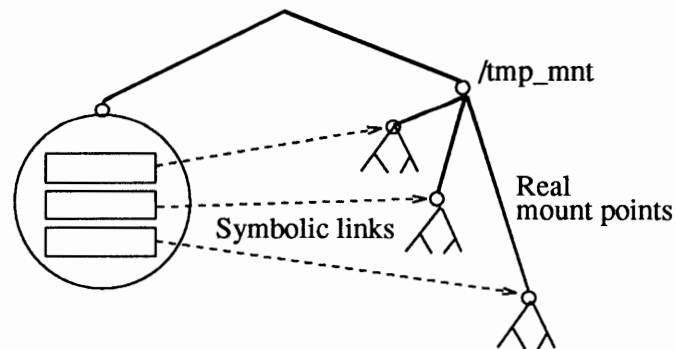
- If the name is found in a direct map, the automounter emulates a symbolic link, as stated above. It responds as if a symbolic link exists at its mount point. In response to a GETATTR, it describes itself as a symbolic link. When the kernel follows up with a READLINK it returns a path to the *real* mount point for the remote hierarchy in `/tmp_mnt`:

Figure 15-11 Automount's Symbolic Link: Direct Map



- If, on the other hand, the name is found in an indirect map, the automounter emulates a directory of symbolic links. It describes itself as a directory. In response to a READLINK, it returns a path to the mount point in `/tmp_mnt`, and a `readdir(3)` of the automounter's mount point returns a list of the entries that are *currently* mounted:

Figure 15-12 Automount's Symbolic Link: Indirect Map



Whatever the case, that is, whether the map is direct or indirect, if the file hierarchy is already mounted and the symbolic link has been read recently, the cached symbolic link is returned immediately. Since the automounter is on the same host, the response is much faster than a READLINK to a remote NFS server. On the other hand, if the file hierarchy is not mounted, a small delay will occur while the mounting takes place.

Error Messages Related to automount

The following paragraphs are labeled with the error message you are likely to see if the automounter fails, and an indication of what the problem may be.

Error Messages Generated by the Verbose Option

- `no mount maps specified`

The automounter was invoked with no maps to serve, and it cannot find the NIS `auto.master` map. It exits. Recheck the command, or restart NIS if that was the intention.

- `mapname: Not found`

The required map cannot be located. This message is produced only when the `-v` option is given. Check the spelling and pathname of the map name.

- `leading space in map entry entry text in mapname`

The automounter has discovered an entry in an automount map that contains leading spaces. This is usually an indication of an improperly continued map entry e.g.

```
foo
  /bar      frobz:/usr/frotz
```

In the example above, the warning is generated when the automounter encounters the second line, because the first line should be terminated with a backslash (`\`).

- `bad key key in indirect map mapname`

While scanning an indirect map the automounter has found an entry key containing a `/`. Indirect map keys must be simple names - not pathnames.

- `bad key key in direct map mapname`

While scanning a direct map the automounter has found an entry key without a prepended `/`. Keys in direct maps must be full pathnames.

- `NIS bind failed`

The automounter was unable to communicate with the `ybind` daemon. This is information only - the automounter will continue to function correctly provided it requires no explicit NIS support. If you need NIS, check to see if there is a `ybind` daemon running.

- `Couldn't create mountpoint mountpoint: reason`

The automounter was unable to create a mountpoint required for a mount. This most frequently occurs when attempting to hierarchically mount all of a server's exported file systems. A required mountpoint may exist only in a file system that cannot be mounted (it may not be exported) and it cannot be created because the exported parent file system is exported read only.

- `WARNING: mountpoint` already mounted on
The automounter is attempting to mount over an existing mountpoint. This is indicative of an internal error in the automounter (a bug).
- `server:pathname` already mounted on `mountpoint`
The automounter is attempting to mount over a previous mount of the same file system. This could happen if an entry appears both in `/etc/fstab` and in an automounter map (either by accident or because the output of `mount -p` was redirected to `fstab`). Delete one of the redundant entries.
- `can't mount server:pathname: reason`
The mount daemon on the server refuses to provide a file handle for `server:pathname`. Check the export table on `server`.
- `remount server:pathname on mountpoint: server not responding`
The automounter has failed to remount a file system it previously unmounted. This message may appear at intervals until the file system is successfully remounted.
- `WARNING: mountpoint` not empty!
The mount point is not an empty directory. The directory `mountpoint` contains entries that will be hidden while the automounter is mounted there. This is advisory only.

General Error Messages

- `pathok: couldn't find devid device id`
An internal automounter error (bug)
- `WARNING: default option "option" ignored for map mapname`
Where `option` is an unrecognized default mount option for the map `mapname`.
- `option` ignored for `key` in `mapname`
The automounter has detected an unknown mount option. This is advisory only. Correct the entry in the appropriate map.
- `bad entry in map mapname "key"`
- `map mapname, key key: bad`
The map entry is malformed, and the automounter cannot interpret it. Recheck the entry; perhaps there are characters in it that need escaping.
- `Can't get my address`
The automounter cannot find an entry for its host in `NIS/etc/hosts`(or `hosts.byname`).

- Cannot create UDP service
Automounter cannot establish a UDP connection.
- `svc_register` failed
Automounter cannot register itself as an NFS server. Check the kernel configuration file.
- `couldn't create pathname: reason`
Where *pathname* is `/tmp_mnt` or the argument to the `-M` command line option.
- `Can't mount mountpoint: reason`
The automounter couldn't mount its daemon at *mountpoint*
- `Can't update pathname`
Where *pathname* is `/etc/mstab` it means that the automounter was not able to update the mount table. Check the permissions of the file.
- `exiting`
This is an advisory message only. The automounter has received a SIGTERM (has been killed) and is exiting.
- `WARNING: pathname: line line number: bad entry`
Where *pathname* is `/etc/mstab` it means that the automounter has detected a malformed entry in the `/etc/mstab` file.
- `server:pathname` no longer mounted
The automounter is acknowledging that *server:pathname* which it mounted earlier has been unmounted by the `umount` command. The automounter will notice this within 1 minute of the unmount or immediately if it receives a SIGHUP.
- `trymany: servers not responding: reason`
No server in a replicated list is responding. This may indicate a network problem.
- `server:pathname - linkname : dangerous symbolic link`
The automounter is attempting to use *server:pathname* as a mountpoint but it is a symbolic link that resolves to a *pathname* referencing a mount point outside of `/tmp_mnt` (or the mount point set with the `-M` option). The automounter refuses to do this mount because it could cause problems in the host's file system, e.g. mounting on `"/usr"` rather than in `"/tmp_mnt"`.
- `host server` not responding
The automounter attempted to contact *server* but received no response.

- `Mount of server:pathname on mountpoint: reason`
The automounter failed to do a mount. This may indicate a server or network problem.
- `pathconf: server: server not responding`
The automounter is unable to contact the mount daemon on *server* that provides (POSIX) pathconf information.
- `pathconf: no info for server:pathname`
The automounter failed to get pathconf information for *pathname*.
- `hierarchical mountpoints: pathname1 and pathname2`
The automounter does not allow its mountpoints to have a hierarchical relationship i.e. an automounter mountpoint must not be contained within another automounted file system.
- `mountpoint: Not a directory`
The automounter cannot mount itself on *mountpoint* because it's not a directory. Check the spelling and pathname of the mount point.
- `dir mountpoint must start with '/'`
Automounter mount point must be given as full pathname. Check the spelling and pathname of the mount point.
- `mapname: yp_err`
Error in looking up an entry in an NIS map. May indicate NIS problems.
- `hostname: exports: rpc_err`
Error getting export list from *hostname*. This indicates a server or network problem.
- `nfscast: cannot send packet: reason`
The automounter cannot send a query packet to a server in a list of replicated file system locations.
- `nfscast: cannot receive reply: reason`
The automounter cannot receive replies from any of the servers in a list of replicated file system locations
- `nfscast:select: reason`
- `Cannot create socket for nfs: rpc_err`
All these error messages indicate problems attempting to ping servers for a replicated file system. This may indicate a network problem.

- NFS server (*pidn@mountpoint*) not responding still trying

An NFS request made to the automount daemon with PID *n* serving *mountpoint* has timed out. The automounter may be temporarily overloaded or dead. Wait a few minutes; if the condition persists, the easiest solution is to reboot the client. If not, you have to exit all processes that use automounted directories (or, change to a non automounted directory in the case of a shell), kill the current automount process and restart it again from the command line. If this does not work, you have to reboot.

The Network Information Service

16.1. What is NIS

The Network Information Service (NIS) was formerly known as Sun Yellow Pages. The functionality of the two remains the same, only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications plc and may not be used without permission.

There exist however some differences between previous releases of this service and the one in Release 4.1 of the SunOS operating system.

NIS is a distributed name service designed to meet the administrative needs of diverse and evolving computing communities. It is a mechanism for identifying and locating objects and resources accessible to the community. It provides a uniform storage and retrieval method for network-wide information in a transport protocol- and media-independent fashion.

By running the NIS service, the system administrator can distribute administrative databases (maps) among a variety of machines, and update those databases from a centralized location in an automatic and reliable fashion, thus ensuring that all clients share the same databases in a consistent manner throughout the network. Furthermore, the use of the NIS “publickey” map permits running secure RPC and secure NFS across the network of machines.

The NIS elements

The NIS service is composed of the following elements:

- domains
- maps
- daemons:
 - `ypserv` — server process
 - `yplibind` — binding process
 - `ypxfrd` — high speed map transfer daemon
 - `rpc.ypupdated` — server for changing map entries
 - `rpc.yppasswdd` — server for modifying NIS password file
 - `in.named` — optional (see *Administering Domain Name Service*)
- utilities:

- `ypcat` — list data in a map
- `ypwhich` — list name of NIS server and maps served
- `ypmatch` — match a key to its value in a map
- `ypinit` — build and install an NIS database.
- `yppoll` — get a map's order number from a server
- auxiliary utilities:
 - `yppush` — propagate data from master to slave NIS server
 - `ypset` — set binding to a particular server
 - `ypxfr` — transfer data from master to slave NIS server
 - `makedbm` — create dbm file for an NIS map

This chapter explains how to set up and administer the NIS name service. Information in the chapter includes:

- The NIS environment, which explains what is an NIS domain, what are NIS servers and clients, and what are the NIS maps.
- Deploying the NIS service, which explains how to establish an NIS domain, how to prepare files on clients and servers, how to make the maps, how to set up servers and clients, and how to propagate the maps.
- Administering Users on an NIS Network, which explains how to add new users to an NIS domain and what are some of the security implications of running NIS.
- Obtaining Map Information, which lists some methods by which users can browse the information contained in NIS maps.
- Advanced NIS Administration, which explains how to update NIS maps, how to modify the Makefile, how to add maps, how to move master maps to another server, and how to use NIS together with the Domain Name Service.
- Summary of NIS Related Commands
- Fixing NIS Problems
- Turning Off NIS Services
- Default NIS Maps

16.2. The NIS Environment

NIS service is provided by the daemons `ypserv` and `ypbind`, and updates are handled by the daemon `rpc.yupdated`.

The NIS service uses information contained in NIS *maps*. This information usually derives from text files such as those traditionally found in the `/etc` directory. Each NIS map has a *mapname*. On a network running NIS, *at least one* NIS server per domain (see below) maintains a set of NIS maps for other hosts in the domain to query.

The NIS Domain

A *NIS domain* shares a common set of maps. Each domain has a *domainname*. Each machine belongs to a domain; machines belonging to the same domain share the same maps.

No restrictions are placed on whether a machine can belong to a given domain, as long as there is a server (see below) for that domain's maps in the same network. If the machine is in a subnet (see the chapter *The SunOS Network Environment*) there must be a server within the subnet.

Note: The domain name is case sensitive.

Assignment to a domain is done at the local level of each machine through the command `domainname`, as seen below.

NIS Machine Types

There are three types of NIS machines:

- master server
- slave server
- client

Any machine can be an NIS client, but only machines with disks should be NIS servers, either master or slave. Servers are also clients.

NIS servers

By definition, an NIS server is a machine with a disk storing a set of NIS maps that it makes available to network hosts. The NIS server does not have to be the same machine as the file server, unless, of course, it is the only machine on the network with a disk. Preferably, also, no HP server is a gateway machine, since this can lead to some problems under heavy load conditions.

NIS servers come in two varieties, master and slave. The machine designated as NIS *master server* contains the set of maps that you, the NIS administrator, update as necessary. If there is only one NIS server on the network, designate it as the master server. Otherwise, designate the machine that will propagate NIS updates with the least performance degradation. In all cases, the master server should be running release 4.x if there is a mixture of machines running 3.x and 4.x releases in the network. We recommend that the master server run release 4.1 because of the new `ypxfrd` daemon to speed map transfers.

You can designate additional NIS servers on the network as *slave servers*. A slave server has a complete copy of the master's set of NIS maps: Whenever the master server's maps are updated, it propagates the updates among the slave servers. The existence of slave servers allows the system administrator to distribute evenly the load implied in answering NIS requests. It also minimizes the impact of a server becoming unavailable. The following diagram is a stylized representation of the relationship between master, slaves and clients, where the arrows represent the possible client-server binding at any one time:

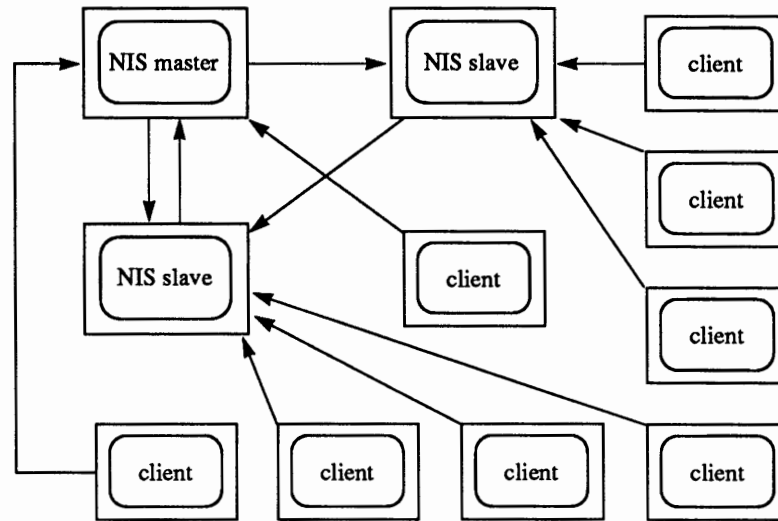


Figure 16-1 *Master, slaves and clients*

A server may be a master in regard to one map, and a slave in regard to another. However, randomly designating a server as master of one map and another server as master of another map can cause a great deal of administrative confusion. You are strongly urged to make a single server the master for all the maps you create within a single domain. The examples in this chapter assume that one server is the master for all maps in the domain.

NIS Clients

NIS clients run processes that request data from maps on the servers. Clients do not care which server is the master in a given domain, since all NIS servers should have the same information. The distinction between master and slave server only applies to *where* you make the updates.

NIS Binding

NIS clients get information from the NIS server through the binding process, which does the following:

1. A program running on the client (that is, a client process) needing information provided by an NIS map, asks `ypbind` for the name of a server.
2. The function of `ypbind` is to remember the address at which the `ypserv` process is listening for requests. `ypbind` sends an RPC broadcast message for `ypserv` processes and binds to the first one to respond. This information is cached in the file `domainname.version` in the directory `/var/yp/binding` (see below, subsection *Establishing the Domain*.)
3. `ypbind` tells the client process to which server to talk. The client sends the request directly to the server.
4. The `ypserv` daemon on the NIS server handles the request by consulting the appropriate map, and

5. it sends the requested information back to the client.

The binding between a client and a server can change in accordance with the network's load, as the service tries to compensate for the current activity; that is, a client may get its information from one server at one time and from another server at a different time.

To find out which NIS server is currently providing service to a client, use the `ypwhich` command as follows:

```
% ypwhich [hostname]
```

where *hostname* is the name of the client. If no `hostname` is mentioned, `ypwhich` defaults to the local host (that is, the machine on which the command is entered.)

ypbind and Subnetting

The client process, as seen above, uses an RPC broadcast to initiate a binding. Since broadcasts are only local subnet events that are not routed further, there must be at least one domain server (master or slave) on the same subnet as the client. The domain servers themselves may exist throughout different subnets since domain map propagation works across subnet boundaries. In a subnet environment, one common method is, if possible, to make the subnet router an NIS domain server. This allows it to serve clients on either subnet interface.

NIS Maps

NIS maps are one type of implementation of SunOS databases. (Other types, not necessarily overlapping, are the files generally found in the `/etc` directory and the DNS files.)

Maps for each domain are located in a separate, distinct directory, `/var/yp/domainname` on the NIS server. That is, the maps for machines that belong to the domain `accounting` will be located in the directory `/var/yp/accounting` on their corresponding NIS server.

Information in NIS maps is organized in the format of `ndbm` files.

Note: Never make the maps on a slave server.

SunOS Release 4.1 supplies a Makefile in the directory `/var/yp` of machines designated as master servers at installation time. Running `make` in that directory (as explained below in *Making the maps*) causes `makedbm` to create the default NIS maps from the input files. The last section of this chapter provides a complete listing and explanation of these maps.

The `ypfiles(5)` and `ndbm(3)` man pages completely explain the `ndbm` file format. Input to `makedbm` must be in the form of *key/value* pairs, where *key* is the first word of each line and *value* is whatever follows in that line. The input can come directly from a file or from standard input (as when modified through a script; see below, *Making the maps*). After passing through `makedbm` the data is collected in two files, `mapname.dir` and `mapname.pag`, both in the `/var/yp/domainname` directory on the master server.

Table 16-1 *A Basic Set of NIS Maps*

bootparams	netid.byname
ethers.byaddr	netmasks.byaddr
ethers.byname	networks.byaddr
group.bygid	networks.byname
group.byname	passwd.byname
hosts.byaddr	passwd.byuid
hosts.byname	protocols.byname
mail.aliases	protocols.bynumber
mail.byaddr	publickey.byname
netgroup.byhost	rpc.bynumber
netgroup.byuser	services.byname
netgroup	ypservers

You may want to use all these maps or only specific maps. NIS can also serve any number of maps you create or add when you install other software products.

NIS services make updating network databases much simpler. You no longer have to change the administrative files in `/etc` on every machine each time you modify the network environment. For example, when you add a new machine to a network without NIS, you must update `/etc/hosts` on every machine on the network. With NIS, you only have to update the input file in the master server and run `make`. This automatically updates the `hosts.byname` and `hosts.byaddr` maps, among others. These maps are then transferred to other servers, and are then available to all the clients. Then programs that previously consulted `/etc/hosts` send a remote procedure call to the NIS servers for the same information. The NIS server refers to the `hosts.byname` and `hosts.byaddr` maps, then sends the requested information to the client.

You can use the `ypcat` command to display the values in a map. Its basic format is

```
ypcat mapname
```

where *mapname* is the name of the map you want to examine or its nickname. To obtain a list of available map nicknames, such as `passwd` for `passwd.byname`, enter `ypcat -x` or `ypwhich -x`. The nicknames are hard-coded, and can be neither modified nor added to. If a map is composed only of keys, as in the case of `ypservers`, use `ypcat -k`; otherwise `ypcat` will print blank lines. The `ypcat(1)` man page describes more options for `ypcat`.

You can use the `ypwhich` command introduced earlier to find out which server is the master of a particular map. Type the following:

```
% ypwhich -m map.name
```

where *map.name* is the name or the nickname of the map whose master you want

to find. `ypwhich` responds by displaying the name of the server. For complete information, refer to the `ypwhich(8)` man page.

16.3. Deploying the NIS Service

Deploying the NIS service consists of the following steps:

- (1) Establishing the domain(s) for your machines
- (2) Preparing network databases for NIS service
- (3) Preparing the files on NIS clients
- (4) Preparing network databases on the master server
- (5) Making the maps
- (6) Setting up the master server
- (7) Setting up slave servers
- (8) Setting up NIS clients
- (9) Modifying default maps
- (10) Propagating NIS maps

The following sections explain each of the above steps.

Establishing the Domain

Before you configure machines as NIS servers or clients, you must prepare the NIS domain by:

- Giving it a name

A domain name can be up to 256 characters long, though you should limit yourself to much shorter names, with a practical upper limit of 32 characters.

- Designating which machines will serve or will be served by the NIS domain.

Once you have chosen a domain name, make a list of network hosts that will give or receive NIS service within that domain.

- Determine which machine should be master server (you can always change this at a later date.)
- List which hosts on the network, if any, are to be slave servers.
- Finally, list all the hosts that are to be NIS clients.

You probably will want all hosts in your network's administrative domain to receive NIS services, although this is not strictly necessary. If this is the case, give the NIS domain a name within the network administrative domain. For instance, in the Internet domain `widget.com`, you could call it `yp.widget.com`. You can also have more than one NIS domain name within the Internet domain. Refer back to the chapter *The SunOS Network Environment*, and forward to the chapter *Administering Domain Name Service* for a full description of network domain names.

Before a machine can use NIS services, its correct NIS domain name and host name must be set. Except in the case of diskless machines, which receive the

appropriate information at boot time from their server, the host name is taken from a file in `/etc` called `hostname.xx#`, where `xx#` stands for a network interface, such as `le0` or `ie0`, and the domain name is taken from the file `/etc/defaultdomain`.

`SunInstall` automatically updates these files for every machine you have it configure. Thereafter, when these machines boot, their host names and NIS domain names are set through `/etc/rc.boot` and `/etc/rc.local` respectively, where the above mentioned files are consulted.

However, if you want to change the NIS domain name or host name on a machine, log in as superuser, and edit the `/etc/defaultdomain` file, changing its present content with the new domain name for the machine. To change the host name, edit the appropriate `hostname.xx#` file (there will be more than one if the machine has more than one network interface), and replace the old host name with the new one. Reboot the machine: After changing the domain name or the hostname you must reboot it, because some library routines may have cached the old domainname and/or hostname, and it is the only way of changing the cache.

Preparing Network Databases for NIS Service

Until NIS is deployed, each host accesses network information through the local `/etc` files. After NIS is set up, hosts should obtain information from the NIS maps, rather than from the local files. For this to happen the local administrator of each host, or the system administrator, must abbreviate or eliminate the files in `/etc` that represent the same databases as the NIS maps. Files on the master server should also be prepared. These files are:

```
passwd
hosts
ethers
group
networks
protocols
services
aliases
netmasks
```

Preparing Files on NIS Clients

Here are the changes, if any, that have to be made to `/etc` files on all NIS clients. These changes should be made either by the system administrator through the file server, or by the local administrator of each host. Note that the files `networks`, `protocols`, `ethers`, and `services` need not be present on any NIS clients. However, if a client will on occasion not run NIS, make sure that the above mentioned files do have valid data in them.

Preparing `hosts.equiv`

The `hosts.equiv` file, which is fully described in *The SunOS Network Environment*, does not have an equivalent NIS map. However, either the system administrator or the local administrator can add escape sequences to it that reference the NIS hosts database.

Note: This is unsafe if there are

untrusted hosts in the network.

If `hosts.equiv` contains a single line with only the character `+` (plus) in it, then anyone listed in the password database (whether in the local `passwd` file or, if it contains a `+`, in the NIS map) can access the machine from any known host. This is the default.

Designating trusted groups allows for more control over logins. Trusted groups are described in the chapter *Administering Security*. NIS assumes that the trusted group name appearing after the `+@` or `-@` sign is a netgroup defined in the `netgroups` map. Refer to *The SunOS Network Environment* for more information.

If a machine's `hosts.equiv` does not have escape sequences, only by the entries in the file determine granting remote access, and NIS maps are not consulted.

Preparing `.rhosts`

`.rhosts` also does not have an equivalent NIS map. *The SunOS Network Environment* fully discusses its format and restrictions. Make the list of trusted hosts explicit, or use netgroup names for the same purpose. You can not use secondary hostnames (nicknames) in your `.rhosts` or `hosts.equiv` files. You can, however, use secondary hostnames in `/etc/hosts`. All of the above files are related in that they enable remote machines to access local machines in some fashion.

Preparing `hosts`

Even when set up for receiving NIS service, the client's `/etc/hosts` file must contain entries for the local host's name, and the local loopback name. If the client is a diskless machine, it also must contain the address of its file server. The `/etc/hosts` file is accessed at boot time before the client's `ybind` daemon starts up. Once `ybind` is running, the local `/etc/hosts` file is not accessed at all. Rather, information is obtained from the `hosts.byaddr` and `hosts.byname` maps. Refer to *The SunOS Network Environment* for more information on `/etc/hosts`.

Preparing `passwd`

Programs first consult an NIS client's local `/etc/passwd` file to determine access permission before consulting the NIS maps. Therefore, every client's `/etc/passwd` should contain entries for root and the primary users of the machine. `/etc/passwd` should also have the `+` escape entry to force the use of the `passwd.byname` and `passwd.byuid` NIS maps. If there is no `+` entry, programs will not consult the NIS maps at all.

A sample NIS client's `/etc/passwd` file looks like:

Check the following input files on the master server to make sure they reflect an up-to-date picture of your system:

```
passwd
hosts
ethers
group
networks
protocols
services
bootparams
netgroup
```

An entry for the daemon user ID must be present in the local `passwd` file for both master and slave server. Furthermore, that entry must precede any other entries with the same user ID, as described previously for setting up the client's `/etc/passwd` file.

Also, make sure the `aliases` input file is complete by verifying that it contains all the mail aliases that you want to have available *throughout* the domain. Refer to the chapter on *Administering Electronic Mail*, and the `aliases(5)` man page for more information.

The `publickey` map

If you are planning on running secure RPC or secure NFS as a means of providing secure networking, you need to prepare the `publickey` file.

This file consists of three fields in the following format:

```
user name      user's public key : user's secret key
```

where *user name* may be the name of a user or of a machine, *user's public key* is that key in hexadecimal notation, and *user's secret key* is that key also in hexadecimal notation, encrypted by the user's password.

Since nobody expects you to be conversant in hexadecimal notation, the program `newkey` is provided to make things easier. The program creates a new public/secret key pair in the `publickey` file, encrypted with the login password of the given user.

Users can later on modify their own entries, or can even create them, by using the program `chkey`.

For instructions on the use of `newkey` and `chkey`, consult their respective man pages in the *SunOS Reference Manual*, and the section on *Secure NFS* in the chapter *The Sun Network File System Service*.

Note that in order for `newkey` and `chkey` to be able to run properly, the daemon `rpc.yupdated` must be running on the master server. If it is not running at this point, enter

```
# rpc.yupdated
```

and also make sure that the master server's `/etc/rc.local` file contains the lines

```

umask 022
if [ -f /usr/etc/rpc.yupdated -a -d /var/yp/`domainname` ]; then
    rpc.yupdated;      echo -n ' yupdated'
fi

```

The `yupdated` daemon consults the file `/var/yp/updaters` for information about which maps should be updated and how to go about it. In the case of the `publickey.byname` map, changes to the `publickey` text file affected through `newkey` or `chkey` are mediated by `/usr/etc/yp/udppublickey`.

`updaters` is a make file, and it is installed by default with an entry for `/etc/publickey`. If you modify `publickey`'s directory in `/var/yp/Makefile` (that is, if you want the `publickey` file to be in a directory other than `/etc`) you should also modify the variable `DIR` in `updaters` in a manner similar to that specified below in *Modifying the Makefile*.

Other Maps

Most maps, unlike `publickey.byname`, do not need the assistance of special programs for their creation or modification. For instance, if you are planning on having distributed automounter files (for more information on the automounter, see the chapter *Using the NFS Automounter*) all you have to do is write the automounter files as they would reside in a machine's `/etc` directory, but make sure that the notation `+mapname` at the beginning of a line, as an indication to consult the corresponding NIS map, is not present in the input files that you prepare for `makedbm` on the master server.

Making the maps

The next step, after creating the input files, is to convert them into the `dbm` format that the NIS service expects. This is done automatically for you by `ypinit` when you call it on the master server, as explained below, in *Setting Up the Master Server*. `ypinit` (among other things) calls the program `make`, which uses the `Makefile` located in `/var/yp`.

The default Makefile

A `Makefile` is provided for you in the directory `/var/yp`. It contains the commands needed to transform the input files into the desired `dbm` format. Unless you add non-default maps, or change the value of `DIR` in the `Makefile`, as explained later, you will not need to ever modify it. The `Makefile` contains commands for `make`, and is similar to the following:


```

#
#  @(#)make.script 1.30 89/05/09 smi
#
# Set the following variable to "-b" to have yp servers use the domain name
# resolver for hosts not in the current domain.
#B=-b
B=
DIR =/etc
DOM = `domainname`
.
.
all: passwd group hosts ethers networks rpc services protocols \
    netgroup bootparams aliases publickey netid netmasks c2secure
.
.
passwd.time: $(DIR)/passwd
    @(awk 'BEGIN { FS=":"; OFS="\t"; } /^[a-zA-Z0-9_]/ { print $$1, $$0 }' $(DI
R)/passwd $(CHKPIPE)) | $(MAKEDBM) - $(YPDBDIR)/$(DOM)/passwd.byname;
    @(awk 'BEGIN { FS=":"; OFS="\t"; } /^[a-zA-Z0-9_]/ { printf("%-10d ", $$3);
print $$0 }' $(DIR)/passwd $(CHKPIPE)) | $(MAKEDBM) - $(YPDBDIR)/$(DOM)/passwd.b
yuid;
    @touch passwd.time;
    @echo "updated passwd";
    @if [ ! $(NOPUSH) ]; then $(YPPUSH) -d $(DOM) passwd.byname; fi
    @if [ ! $(NOPUSH) ]; then $(YPPUSH) -d $(DOM) passwd.byuid; fi
    @if [ ! $(NOPUSH) ]; then echo "pushed passwd"; fi

group.time: $(DIR)/group
    @(awk 'BEGIN { FS=":"; OFS="\t"; } { print $$1, $$0 }' $(DIR)/group $(CHKPI
PE)) | $(MAKEDBM) - $(YPDBDIR)/$(DOM)/group.byname;
    @(awk 'BEGIN { FS=":"; OFS="\t"; } { printf("%-10d ", $$3); print $$0 }' $(
DIR)/group $(CHKPIPE)) | $(MAKEDBM) - $(YPDBDIR)/$(DOM)/group.bygid;
    @touch group.time;
    @echo "updated group";
    @if [ ! $(NOPUSH) ]; then $(YPPUSH) -d $(DOM) group.byname; fi
    @if [ ! $(NOPUSH) ]; then $(YPPUSH) -d $(DOM) group.bygid; fi
    @if [ ! $(NOPUSH) ]; then echo "pushed group"; fi
.
.
passwd: passwd.time
group: group.time
hosts: hosts.time
ethers: ethers.time
networks: networks.time
.
.
netmasks: netmasks.time
$(DIR)/netid:

```

The function of the Makefile is to create, for each of the databases listed under all, the appropriate maps for NIS to use. In most cases the input file must first be cleaned of comment and other extraneous lines through some sed or awk

script. Then the fields of the file must be ordered correctly for `makedbm`. This is very clear in the case of `passwd`. In order to create the map `passwd.byname` `makedbm` must be passed the *key/value* pair of `name/entry` while in order to create the `passwd.byuid` the pair must be `uid/entry`. This is accomplished by the `awk` script that in the first case passes to `makedbm` the first field of the password file and then the whole entry, and in the second case passes the third field followed, again, by the whole entry.

Setting Up the Master Server

The `/usr/etc/yp/ypinit` shell script helps you establish the master and slave servers. It also runs `make` initially to create the maps, and starts their propagation.

If servers in the network are running release 3.x and 4.x of the SunOS operating system, always choose a server running 4.x to be your master server. A server running 4.1 is in fact to be preferred over one running previous releases because of performance improvements.

You use `ypinit` to build a fresh set of NIS maps on the master server in the following way:

1. Boot the machine that is going to be your master server in single-user mode, and, as superuser:
2. Type

```
# cd /var/yp
# /usr/etc/yp/ypinit -m
```

3. `ypinit` asks whether you want the procedure to terminate at the first non-fatal error or continue despite non-fatal errors.

If you choose the first option, `ypinit` will exit upon encountering the first problem; you can then fix it and restart `ypinit`. This is recommended if you are running `ypinit` for the first time. If you prefer to continue, you can try to fix by hand all problems that may occur, then restart `ypinit`.

4. `ypinit` prompts for a list of other hosts to become NIS servers. Enter the name of all other NIS servers.
5. Once `ypinit` has constructed the list of servers, it start `make(1)`. This program uses the instructions contained in the Makefile file (either the default one or the one you modified) located in `/var/yp`: it cleans all comment lines from the files you designated and runs `makedbm` on them, creating the appropriate maps and establishing the name of the master server for each map.

Note that if the map or maps being pushed by the Makefile correspond to a domain other than the one returned by the command `domainname` on the master, you can make sure that they are pushed to the correct domain by starting `make` in the `ypinit` shell script with a proper identification of the variable `DOM`:

```
make DOM=correct_domain passwd
```

will push the password map to the intended domain instead of the domain to which the master belongs.

For security reasons, you may want to restrict access to the master NIS machine to a smaller set of users than that defined in its `/etc/passwd` file. To do this, copy the complete file and give the copy a name and path other than `/etc/passwd`. Edit out undesired users from the remaining `/etc/passwd`. For a security-conscious system, this smaller file should not include the NIS escape entry (+).

Creating the Master Server

Now that the master maps are created, you can create the master server and begin NIS service. To do this, you have to run `ypinit` again, then start up `ypserv` on the server. When a client requests information from the server, `ypserv` is the daemon that answers information requests from clients after looking it up in the NIS maps.

1. From `/var/yp`, type the following

```
# /usr/etc/yp/ypinit -m
```

to set up the master server.

2. Next, type

```
# ypserv
```

to start providing NIS services. The next time you boot the server, `ypserv` will automatically start up from `/etc/rc.local`.

3. Look for the following lines on the master server's `/etc/rc.local` file:

```
if [ -f /usr/etc/ypserv -a -d /var/yp/`domainname` ]; then
    ypserv;                echo -n ' ypserv'
#
#   Master NIS server runs the xfr daemon
#
#   ypxfrd;                echo -n ' ypxfrd'
fi
```

and uncomment the line that says:

```
#   ypxfrd;                echo -n ' ypxfrd'
```

4. Finally, type

```
# ypxfrd
```

to start the high speed transfer daemon (always do this only on the master server.) The next time you boot the master server, `ypxfrd` will

automatically start up from `/etc/rc.local`.

Setting Up Slave Servers

Your network can have one or more NIS slave servers. Having slave servers ensures the continuity of NIS services in the event of the master server being down. Before actually running `ypinit` to create the slave servers, you should take several precautions.

The domain name for each NIS slave must be the same as that for the NIS master server. Use the `domainname` command on each NIS slave to make sure they are consistent with the master server. Make any changes to the domain name as necessary, as described in the previous section, *Establishing the Domain*. Also, do not forget to set each slave server's correct host name.

As you did with the NIS master server, you must also check that every slave server's password database contains an entry for the daemon user name and that it precedes other entries with the same user ID.

Make sure that the network is working properly before you set up a slave NIS server. In particular, check that you can use `rccp` to send files from the master NIS server to NIS slaves. If you cannot, follow the procedures outlined in *The SunOS Network Environment* to permit the use of `rccp`.

Now you are ready to create a new slave server.

1. Reboot each slave server in single user mode.
2. Change directory to `/var/yp`.
4. Type the following:

```
# ypbind
```

5. Type the following:

```
# /usr/etc/yp/ypinit -s master
```

where *master* is the hostname of the existing NIS master server.

`ypinit` will not prompt you for a list of other servers, as it does when you create the master server. However, it does let you choose whether to halt at the first non-fatal error. `ypinit` then calls the program `ypxfr`, which transfers a copy of the master's NIS map set to the slave server's `/var/yp/domainname` directory.

6. When `ypinit` terminates, make backup copies of the `/etc/passwd` and `/etc/group` files, and edit the original files by adding `+:` to the end so they refer to the NIS maps. This applies also to whatever non-default maps you may have added to the Makefile. For instance, if there is an `auto.direct` NIS map, the file `/etc/auto.direct` should also be copied to another file and the original should contain a line similar to

```
+auto.direct
```

Thus, whenever the automounter reads this file, it will consult the NIS `auto.direct` map upon reaching this line.

The above procedures ensure that processes on the slave server also use the NIS services, rather than files in the local `/etc`.

7. Type

```
# ypserv
```

to begin NIS services on the slave server. The next time you reboot the NIS slave, `ypserv` will start automatically from `/etc/rc.local`.

Repeat the procedures above for each machine you want configured as an NIS slave server.

Setting Up NIS clients

1. Edit the client's local files, as you did for the local files in the slave servers, so that processes consulting those files are sent to the NIS maps.
2. Type

```
# ps -aux | grep ypbind
```

to confirm that `ypbind` is running. If it is not, enter the command from the command line, and inspect the `/etc/rc.local` file to make sure that the following lines are there, and are not commented out:

```
if [ -d /var/yp ]; then
    if [ -f /etc/security/passwd.adjunct ]; then
        ypbind -s; echo -n ' ypbind'
    else
        ypbind;    echo -n ' ypbind'
    fi
fi
```

At this point, you must have at least one NIS server configured on the network (or on the subnet, if you are subnetting). Otherwise, processes on the client hang if no NIS server is available while `ypbind` is running. If the client is in C2 security mode, `ypbind` will be started with the `-s` option, and will accept bindings only to `ypserv` processes run as root. If none such is available, the situation is the same as having none in the network, and `ypbind` will hang.

3. Add entries for the client in the following maps:

```
ethers
hosts
bootparams
netgroup (If applicable)
```

4. Have users in the client use the `yppasswd` command to create their new passwords in the `passwd` maps. For this to work, the `rpc.yppasswdd`

daemon must be running in the master server. Users can also use `passwd -y` to the same effect, or `passwd` alone if they do not have an entry in the machines' local `/etc/passwd` file.

Notice that running the `passwd` command only affects the local client's environment when the user has an entry in the local `passwd` file. In this case, you must run the `yppasswd` command to change the NIS password maps. Its syntax is:

```
% yppasswd [ login_name ]
```

When a user types the above command, `yppasswd` prompts for the new password twice, as does the local `passwd` command. However, when the user has successfully responded, `yppasswd` (or `passwd` when acting as such) puts the new password in the `passwd.byname` and `passwd.byuid` maps. A user's NIS password can be different from the password on his or her own machine. If the `rpc.yppasswdd` daemon is not set to make and propagate the newly modified password database immediately, there may be a considerable time lag between the time the user sets the new NIS password and the time it takes effect. Even when it is, map pushing is not instantaneous, so if it takes a while it does not mean that the process failed.

Heterogenous Slaves

A problem with the use of a 3.x NIS slave with a 4.x master may initially exist after setting up the 3.x slave. Because the 3.x `ypinit` program does not know of the complete set of 4.x NIS maps, when a 3.x slave is initialized (`ypinit -s master`) the `ypinit` program only runs `ypxfr` for traditional 3.x maps (e.g. `hosts`, `passwd`, `ethers`). This means the NIS slave will not initially have the entire 4.x set of NIS maps (e.g. `publickey`, `netid`, `netmasks`) and consequently the two databases will not be entirely synchronized. For the slave to fully support the NIS domain, the full set of master NIS maps must be transferred to the 3.x slave. To insure that all maps are initially copied to the slave, a script such as the one below can be run after the slave initialization:

```
#!/bin/sh
for MAP in `ypwhich -m | awk '{print $1 }'`
do
/usr/etc/yp/ypxfr -h `ypwhich -m $MAP` $MAP
echo "Transferred $MAP "
done
```

Note: You can modify the `ypinit` shell script to do this automatically.

This script assumes that `ypbind` is running on the slave and is bound to a server on the relevant domain. This script will cause the slave to first inventory all the master maps and then `ypxfr` them across.

Modifying Default Maps

Use the following procedure for updating all *default* maps.

1. Become superuser on the master server. (Always modify NIS maps on the master server.)
2. Edit the input file for the map you want to change, whether that file resides in `/etc` or in some other directory of your choice.
3. Type the following:

```
# cd /var/yp
# make mapname
```

The `make` command will then update your map according to the changes you made in its corresponding file. It will also propagate it among the servers (see below, *Propagating an NIS Map*.)

Do not use this procedure with the `publickey` map. The commands `newkey` and `chkey` (see above, *Preparing Network Databases on the Master Server*) do it all for you.

Note that *making* the map does not propagate it. For this you have to follow the instructions in the following section.

Propagating an NIS Map

When you *propagate* an NIS map, you move it from one server to another—most often from the master to all NIS slave servers. Initially, `ypinit` propagates the maps from master to slaves, as described previously. From then on, you must transfer updated maps from master to slaves by running the `ypxfr` command. You can run `ypxfr` two different ways: periodically through the root `crontab` file; or interactively on the command line.

`ypxfr` handles map transfer in tandem with the `yppush` program. `yppush` should always be run from the master server. The Makefile in the `/var/yp` directory automatically runs `yppush` after you change the master set of maps.

`yppush`'s function is to copy, or *push* a new version of an NIS map from the NIS master to the slave(s). After making a list of NIS servers from the `ypservers` map initially build by `ypinit`, `yppush` contacts each slave server in the list and sends it a “transfer map” request. When the request is acknowledged by the slave, the `ypxfr` program running on the slave does the actual transfer of the new map to the slave.

Using `crontab` with `ypxfr`

Maps have different rates of change. For instance, some may not change for months at a time, such as `protocols.byname` among the default maps and `auto.master` among the non-default, but `publickey` or `passwd.byname` may change several times a day. Scheduling map transfer through the `crontab` command allows setting specific propagation times for individual maps.

To periodically run `ypxfr` at a rate appropriate for the map set, root's `crontab` file on each slave server should contain the appropriate `ypxfr` entries. `ypxfr` contacts the master server and transfers the map only if the master's copy is more recent than the local copy. (Refer to the chapter on *Administering Workstations*, for information about creating `crontab` files through the

crontab command, and see `crontab(1)` in the *SunOS Reference Manual*).

Using Shell Scripts with `ypxfr`

As an alternative to creating separate `crontab` entries for each map, you may prefer to have root's `crontab` run a shell script that periodically updates all maps. There are sample map-updating shell scripts, `ypxfr_1perday`, `ypxfr_1perhour`, and `ypxfr_2perday` in the directory `/usr/etc/yp`. You can easily modify or replace these shell scripts to fit your site's requirements. Here is the default `ypxfr_1perday` shell script:

```
#!/bin/sh
#
# ypxfr_1perday.sh - Do daily yp map check/updates
#

PATH=/bin:/usr/bin:/usr/etc:/usr/etc/yp:$PATH
export PATH

# set -xv
ypxfr group.byname
ypxfr group.bygid
ypxfr protocols.byname
ypxfr protocols.bynumber
ypxfr networks.byname
ypxfr networks.byaddr
ypxfr services.byname
ypxfr ypservers
```

This shell script will update once per day the maps mentioned in it, if root's `crontab` executes it once a day (preferably at times of low network load). You can also have scripts to update maps once a week, once a month, once every hour, etc., but be aware of the performance degradation implied in propagating the maps.

Run the same shell scripts through root's `crontab` on each slave server configured for the NIS domain. Alter the exact time of execution from one server to another to avoid bogging down the master.

If you want to transfer the map from a particular slave server, use the `-h host` option of `ypxfr` within the shell script. Here is the syntax of the commands you put in the script:

```
/usr/etc/yp/ypxfr -h host [ -c ] mapname
```

where *host* is the name of the server with the maps you want to transfer, and *mapname* is the name of the requested map. If you use the `-h` option without specifying *host*, `ypxfr` will try to get the map from the master server. If `ypserv` is not running locally at the time `ypxfr` is executed, you must use the `-c` flag.

You can use the `-s domain` option to transfer maps from another domain to your local domain. These maps should be preferably the same across domains. For

example, two NIS domains might share the same `services.byname` and `services.byaddr` maps. Alternatively, you can use `rcp`, or `rdist` for more control, to transfer files across domains too. See `rdist(1)` in the *SunOS Reference Manual*.

Directly Invoking `ypxfr`

The second method of invoking `ypxfr` is to run it as a command. Typically, you do this only in exceptional situations — for example when setting up a temporary NIS server to create a test environment, or when trying to quickly get an NIS server that has been out of service consistent with the other servers.

Logging `ypxfr`'s Activities

`ypxfr`'s transfer attempts and the results can be captured in a log file. If a file called `/var/yp/ypxfr.log` exists, results are appended to it. No attempt to limit the size of the log file is made. To prevent it from growing indefinitely, empty it from time to time by entering

```
# cp /var/yp/ypxfr.log /var/yp/ypxfr.log.old
# cat /dev/null > /var/yp/ypxfr.log
```

You can have `crontab` execute these commands once a week.

To turn off logging, remove the log file.

16.4. Administering Users on an NIS Network

SunOS Release 4.1 provides a number of features for administering NIS in a secure environment. If your site requires tight security, refer to chapters *The Sun Network File System Service* and *Administering C2 Security*. You may want to use some of the features they discuss, such as secure file systems and C2-like security. If your network requires average security, refer first to *Administering Security*. This next section covers only NIS matters; it assumes you are familiar with the security information in *Administering Security*.

How NIS Maps Affect Security

Security on a system running NIS depends on how programs consult the files in the `/etc` directory that are equivalent to the input files for the NIS maps. A machine's local files are consulted first. These include

```
/etc/passwd
/etc/group
/etc/aliases
```

Then the programs consult maps in the NIS domain that correspond to the local files. For example, a machine checks its own `/etc/aliases` file for mail aliases, then checks the `mail.aliases` NIS map.

The `passwd` file is a good example of how local files take precedence in an NIS environment. When a user runs the `passwd` command to change his or her password, the program first checks if the user has an entry in the local `/etc/passwd` file. If there is no such entry, and NIS is running, and there is a + escape line in the local file, the program acts as `yppasswd` and changes the user's password on the NIS master server for the `passwd` map.

The following files in each machine's `/etc` directory contain network information:

```

/etc/hosts
/etc/networks
/etc/ethers
/etc/services
/etc/netmasks
/etc/protocols

```

However, on a network with NIS, information is obtained not from these local files but rather from their corresponding NIS maps instead, with the following exceptions:

- When booting, each machine needs an entry in `/etc/hosts` for itself and the local loopback name. A diskless machine also needs an entry for its file server.
- Routers (that is, machines with more than one network interface) that are also NIS clients need to have `/etc/netmasks` for use during booting.

Changing the NIS Password

Users must run `passwd -y` or `yppasswd` to change their passwords in the NIS password file (or they can run `passwd` if they do not have an entry in the local `/etc/passwd` file.) Before they can do this, you must start the `rpc.yppasswdd` daemon by adding an entry for it in `rc.local` on the master server for the password map.

The master password file in the master server that is used to build the password maps should not be that server's `/etc/passwd` file. This is because this file should not contain an entry for `root`, and because you may want to limit access to the master server to a small group of people, a goal that would be defeated by having `/etc/passwd` contain entries for all the users. The master password file should therefore be in a directory like `/var/etc`, or any directory of your choice, as long as the file itself is not a link to another file. Go to the master server, edit `rc.local`, and add the following:

```

if [ -f /usr/etc/rpc.yppasswdd -a -d /var/yp/`domainname` ] ; then
    /usr/etc/rpc.yppasswdd /var/etc/passwd -m \
        passwd DIR=/var/yp/domainname
fi

```

where *domainname* is the name of your NIS domain directory. Then reboot the master server to start up the `yppasswdd` daemon. Notice that when the `-m` option is given, a `make` is forced in `/var/yp` immediately following a modification of the file, and the arguments that follow the `-m` are passed over to `make`. If you want to avoid having this `make` take place at any and all times, eliminate it, and control the pushing of the password maps through `crontab`:

```

if [ -f /usr/etc/rpc.yppasswdd -a -d /var/yp/`domainname` ] ; then
    /usr/etc/rpc.yppasswdd /var/etc/passwd
fi

```

The above examples would have to have the name of an adjunct file if you are running a C2 secure network. See the chapter *Administering C2 Security* and the reference to the adjunct file in the man page for `rpc.yppasswdd(8)`.

To actually change the NIS password, have the user type:

```
% yppasswd user_name
```

The system then prompts the user to type the old and new passwords, as the local `passwd` command does. Refer to the `yppasswd(8)` man page for more information about the daemon and to `yppasswd(1)` for information about the `yppasswd` command.

How Netgroups Affect Security on an NIS Network

On a network with NIS, the `netgroup` input file on the master NIS server is used for generating three NIS maps: `netgroup`, `netgroup.byuser` and `netgroup.byhost`. `netgroup` contains the basic information in the `netgroup` input file. The two other NIS maps contain information in a format that speeds lookup of netgroups given the host or user.

Various programs use the `netgroup` NIS maps for permission checking during login, remote mount, remote login, and remote shell creation. These programs include: `mountd`, `login`, `rlogin`, and `rsh`. `login` consults them for user classifications if it encounters `netgroup` names in the `passwd` database. The `mountd` daemon consults them for machine classifications if it encounters `netgroup` names in `/etc/exports`. `rlogin` and `rsh` consult the `netgroup` maps for both machine and user classifications if they encounter `netgroup` names in the `/etc/hosts.equiv` or `/.rhosts` file.

Refer to *The SunOS Network Environment* for more information about `netgroup`.

Adding a New User to an NIS Domain

Adding a new user to a network already running NIS is very similar to adding a new user in a non-networked machine, but it requires a few extra steps and it may imply also adding a new client machine to the NIS maps.

Once the new user's workstation has been added to the network, and the user has been given a password in it as outlined in the chapter *Administering Workstations* and in `adduser(8)`, the first step in adding the new user to the NIS network is to update the `yppasswd` file. Follow these procedures:

1. Log in as `root` on the master NIS server.
2. Edit the master NIS server's `passwd` input file, whether in `/etc` or with another name or in another directory.

The following example shows an initial entry in the NIS master server's password input file:

```
roger::1947:10:The Boss:/home/shams/roger:/bin/csh
```

Note that the fields in the NIS password maps are identical to the local `passwd` file. You should also make sure that the user ID which you assign to this user is unique within the network domain. Failure to keep ID's unique may prevent files on different machines from being moved between directories because the system will respond as if the directories are owned by two

different users. Also, file ownership may become confused when an NFS server exports a directory to an NFS client whose password file contains users with user IDs that match those of different users on the NFS server.

3. After you have updated the master server's password file, update the password NIS maps as follows:

```
# cd /var/yp
# make passwd
```

Note that if your site uses the C2 secure configuration option, it will split `passwd` into two files. In a C2 secure environment, only processes running with superuser privileges can access the file containing the encrypted password.

4. Have the user use the `yppasswd` command to set his or her NIS password.

Making the Home Directory

After you update the NIS password file, and create a password entry for the user on the local machine, create a home directory, as shown in *Administering Workstations*. Note that if you are working on the file server rather than on the user's workstation, and if the NIS password maps have not yet been updated on the machine's NIS server, the following error message appears when you attempt to change ownership of the home directory.

```
unknown user id: username
```

In that case, you can use the following set of commands:

```
# cd /home/servername
# mkdir roger
# chown userid# roger
```

You use the ID number from the password file entry instead of login name to change the ownership of the user's home directory.

16.5. Obtaining Map Information

Users can obtain information from and about the maps at any time by using the commands `ypcat`, `ypwhich` and `ypmatch`. In the examples below, *mapname* refers both to the official name of a map and to its nickname, if any. The examples are not exhaustive, and you should consult the respective man pages in the *SunOS Reference Manual*.

- To list all the values contained by a map, enter:

```
% ypcat mapname
```

- To list both the keys and the values (if any) contained by a map, enter:

```
% ypcat -k mapname
```

- To list all the map nicknames, enter any of:

```
% ypcat -x
% ypwhich -x
% ypmatch -x
```

- To list all the available maps and their master(s), enter:

```
% ypwhich -m
```

- To list the master server for a particular map, enter:

```
% ypwhich -m mapname
```

- To match a key with an entry in a map, enter

```
% ypmatch key mapname
```

- If the item you are looking for is not a key in a map, enter:

```
% ypcat mapname | grep key
```

- To obtain information about other domains, use the `-d` options of these commands.

16.6. Advanced NIS Administration

This section describes how to maintain the maps of an existing NIS domain. Subjects discussed include:

- Updating NIS maps
- Modifying the Makefile
- Adding maps to an additional NIS server
- Moving the master map set to a new server
- Using NIS together with DNS.

Updating Existing Maps

After you have installed NIS, you will discover that some maps require frequent updating while others never need to change. For example, the `publickey.byname` map may change frequently on a large company's network. On the other hand, the `auto.master` map probably will change little, if at all.

When you need to update a map, you can use one of two updating procedures, depending on whether it is a default map or not. A *default* map is a map in the default set created by `ypinit` from the network databases. The section *Modifying Default Maps* above explains how to modify these maps. *non-default* maps may be any of the following:

- Included with an application purchased from a vendor.
- Created specifically for your site.
- Created from a non-text file.

The following explains how to use various updating tools. In practice, you probably will only use them if you add non-default maps or change the set of NIS servers after the system is already up and running.

Modifying the Makefile

You can modify the Makefile provided by default in `/var/yp` to suit your needs. You can delete or add entries, and you can change the names of some of the directories.

Deleting Entries: If you want the Makefile to not produce maps for a specific database, edit it and first delete the name of the database from the `all` rule (that is the line that starts with the word `all`:). Then look for the appropriate rule for that database (that is the rule at the end of the file that, for example for the database `hosts`, says `hosts: hosts.time`) and either delete the line or, just to be in the safe side for the future, comment it out by prepending a `#` to the line.

Adding Entries: To add an entry to the Makefile you have to

- add the name of the database to the `all` rule
- add a rule for the database, and
- write the time rule.

For instance, in order for the `make` file to work on automounter input files, or any map of your own that you wish to propagate through NIS, you would have to make the following modifications to the `make` file:

- (1) modify the line that starts with the word `all`: by adding the name of the database you want to add.

```
all: passwd group hosts ethers networks rpc services protocols \
    netgroup bootparams aliases publickey netid netmasks c2secure \
    auto.direct auto.home
```

The order of the entries is not relevant, but please notice that the blank space at the beginning of the continuation lines *must* be a TAB, not spaces.

- (2) Add the following lines at the end of the `make` file:

```
auto.direct: auto.direct.time
auto.home: auto.home.time
```

- (3) Add an entry for `auto.direct.time` in the middle of the file.

```

auto.direct.time: $(DIR)/auto.direct
@(while read L; do echo $$L; done < $(DIR)/auto.direct $(CHKPIPE) | \
  (sed -e "/^#/d" -e s/#.*$$// -e "/^ *$$/d" $(CHKPIPE)) | \
  $(MAKEDBM) - $(YPDBDIR)/$(DOM)/auto.direct;
@touch auto.direct.time;
@echo "updated auto.direct";
@if [ ! $(NOPUSH) ]; then $(YPPUSH) auto.direct; fi
@if [ ! $(NOPUSH) ]; then echo "pushed auto.direct"; fi

```

The while loop at the beginning is designed to eliminate any backslash-extended lines in the input file. The `sed` script eliminates comment and empty lines, and feeds the output to `makedbm`.

The same procedure should be followed for all other automounter maps, or any other non-default maps.

Changing Variables: You can change the settings of the variables defined at the top of the Makefile simply by changing the value at the right of the equal sign (=). For instance, if you don't use the files located in `/etc` as input for the maps, but rather files located in another directory, say `/var/etc/domainname`, you should change the value of `DIR` from `DIR=/etc` to `DIR=/var/etc/domainname`.

Creating and Modifying Non-Default Maps

To update a non-default map, you edit its corresponding text file. Then you rebuild the updated map using the `/usr/etc/yp/makedbm` program. (The `makedbm(8)` man page fully describes this command.) If the map has an entry in the `/var/yp/Makefile`, simply run `make` as described above. If the map does not have an entry, try to create one following the instructions above, in *Making the maps*. Using `make` is the preferred method. Otherwise, you will have to use `makedbm` by hand.

There are two different methods for using `makedbm`:

- Redirect the program's output to a temporary file, modify the file, then use the modified file as input to `makedbm`.
- Have the output of `makedbm` operated on within a pipeline that feeds into `makedbm`. This is appropriate if you can update the disassembled map with either `awk`, `sed`, or a `cat` `append`.

You can use either of two possible procedures for creating new maps. The first uses an existing text file as input; the second uses standard input.

Updating Maps Built from Existing Text Files

Assume that a text file `/var/yp/mymap.asc` was created with an editor or a shell script on the NIS master. You want to create an NIS map from this file and locate it in the `home_domain` subdirectory. To do this, you type the following on the master server:

```

# cd /var/yp
# /usr/etc/yp/makedbm mymap.asc home_domain/mymap

```

The `mymap` map now exists in the directory `home_domain`.

Adding entries to `mymap` is simple. First, you must modify the text file `mymap.asc`. (If you modify the actual `dbm` files without modifying the corresponding text file, the modifications are lost.) Type the following:

```
# cd /var/yp
# <edit mymap.asc>
# /usr/etc/yp/makedbm mymap.asc home_domain/mymap
```

When you finish updating the map, propagate it to the slave servers, as described in the section *Propagating an NIS Map*.

Updating Maps Built from Standard Input

When no original text file exists, create the NIS map from the keyboard by typing input to `makedbm`, as shown below:

```
ypmaster# cd /var/yp
ypmaster# /usr/etc/yp/makedbm - home_domain/mymap
key1 value1
key2 value2
key3 value3
<ctl D>
ypmaster#
```

If you later need to modify such a map, which is not based in an existing file, you can use `makedbm -u` to disassemble the map and create a temporary text intermediate file. You type the following:

```
% cd /var/yp
% /usr/etc/yp/makedbm -u home_domain/mymap > mymap.temp
```

The resulting temporary file `mymap.temp` has one entry per line. You can edit it as needed, using your preferred editing tools.

To update the map, you give the name of the modified temporary file to `makedbm` as follows:

```
% /usr/etc/yp/makedbm mymap.temp home_domain/mymap
% rm mymap.temp
```

When `makedbm` finishes, propagate the map to the slave servers, as described in the section *Propagating an NIS Map*.

The preceding paragraphs explained how to use some tools, but in reality almost everything you actually have to do can be done by `ypinit` and `/var/yp/Makefile`, unless you add non-default maps to the database, or change the set of NIS servers after the system is already up and running.

Whether you use the `Makefile` in `/var/yp` or some other procedure—`Makefile` is one of many possibilities—the goal is the same: a new pair of well-formed `dbm` files must end up in the maps directory on the master NIS server.

Adding New NIS Maps to the Makefile

Adding a new NIS map entails getting copies of the map's dbm files into the `/var/yp/domain_name` directory on each of the NIS servers in the domain. The actual mechanism is described above in *Propagating an NIS Map*. This section only describes how to update the Makefile so that propagation works correctly.

After deciding which NIS server is the master of the map, modify `/var/yp/Makefile` on the master server so that you can conveniently rebuild the map (as indicated above, different servers can be masters of different maps, but in most cases this can lead to administrative confusion, and it is strongly recommended that you set only one server as the master of all maps). Actual case-by-case modification is too varied to describe here, but typically a human-readable text file is filtered through `awk`, `sed`, and/or `grep` to make it suitable for input to `makedbm`. Refer to the existing `/var/yp/Makefile` for examples and the section *Modifying the Makefile* above.

Use the mechanisms already in place in `/var/yp/Makefile` when deciding how to create dependencies that `make` will recognize. Be aware that `make` is very sensitive to the presence or absence of tabs at the beginning of lines within the dependency rules, and a missing tab can invalidate an entry that is otherwise well formed.

To get an initial copy of the map, you can have `make` run `yppush` on the NIS master server. The map must be globally available before clients begin to access it. If the map is available from some NIS servers but not all, you will encounter unpredictable behavior from client programs.

Adding a New NIS Server to the Original Set

After NIS is running, you may need to create an NIS slave server that you did not include in the initial set given to `ypinit`. If the environment at your site is such that servers are added or removed from the list frequently, you may want to consider adding an entry in the Makefile for the `ypservers` map that was initially created by `ypinit`. After adding the variable `YPSERVERS=$(DIR)/ypservers` and the word `ypservers` to the `all` rule, and adding the rules `ypservers: ypservers.time` and `$(DIR)/ypservers: you should add a rule similar to the following:`

```
ypservers.time: $(YPSERVERS)
    @(awk '!/^#/ && !/^$$/ { print $$1, $$0 }' $(YPSERVERS) $(CHKPIPE) | \
        $(MAKEDBM) - $(YPDBDIR)/$(DOM)/ypservers;
    @touch ypservers.time;
    @echo "updated ypservers";
    @if [ ! $(NOPUSH) ]; then $(YPPUSH) ypservers; fi
    @if [ ! $(NOPUSH) ]; then echo "pushed ypservers"; fi
```

The above is just a suggestion, and you might find some other method more appropriate for your site.

If, however, you do not think it would be justified to modify the Makefile, the following procedure explains how to proceed manually:

1. Log in to the master server as superuser.
2. Go to the NIS domain directory by typing:

```
# cd /var/yp/domain_name
```

3. Disassemble `ypservers`, as follows:

```
# /usr/etc/yp/makedbm -u ypservers > /tmp/temp_file
```

`makedbm` converts `ypservers` from dbm format to the temporary text file `/tmp/temp_file`.

4. Edit `/tmp/temp_file` using your preferred text editor. Add the new slave server's name to the list of servers. Then save and close the file.
5. Run the `makedbm` command with `temp_file` as the input file and `ypservers` as the output file.

```
# /usr/etc/yp/makedbm /tmp/temp_file ypservers
```

This converts `ypservers` back into dbm format.

6. Verify that the `ypservers` map is correct (since there is no text file for `ypservers`) by typing the following:

```
# /usr/etc/yp/makedbm -u ypservers
```

Each entry in `ypservers` will be displayed on your screen.

7. Set up the new slave server's NIS domain directory by copying the NIS map set from the master server. To do this, log in to the new NIS slave as superuser, and run the `ypinit` command:

```
ypslave# cd /var/yp
ypslave# ypbind
ypslave# /usr/etc/yp/ypinit -s ypmaster
```

When you are finished, complete steps 6 through 9 of section *Setting Up Slave Servers*.

Note: If a host name is not in `ypservers` it will not be warned of updates to the NIS map files.

Changing a Map's Master Server

To change a map's master, you first have to build it on the new NIS master. The old master's name occurs as a key-value pair in the existing map (this pair is inserted automatically by `makedbm`). Therefore, using the existing copy at the new master or transferring a copy to the new master with `ypxfr` is insufficient. You have to reassociate the key with the new master's name. If the map has a text source file, you should copy it in its current version to the new master.

Here are instructions for remaking a sample NIS map called `jokes.bypunchline`.

1. Log in to the new master as superuser, and type the following:

```
newmaster# cd /var/yp
```

2. `/var/yp/Makefile` must have an entry for the new map before you specify the map to make. If this isn't the case, edit the `Makefile` now (see above, *Making the maps*.)
3. Type the following:

```
newmaster# make jokes.bypunchline
```

4. If the old master will remain an NIS server, `rlogin` in to it, and edit `/var/yp/Makefile`. Comment out the section of `/var/yp/Makefile` that made `jokes.bypunchline` so that it is no longer made there.
5. If `jokes.bypunchline` only exists as a dbm file, remake it on the new master by disassembling a copy from any NIS server, then running the disassembled version through `makedbm`:

```
newmaster# cd /var/yp
newmaster# ypcat -k jokes.bypunchline | \
/usr/etc/yp/makedbm - mydomain/jokes.bypunchline
```

After making the map on the new master, you must send a copy of it to the other slave servers. However, do not use `yppush`—the other slaves will try to get new copies from the old master, rather than the new one. A typical method for circumventing this (you may find others) is to transfer a copy of the map from the new master back to the old master. Become superuser on the old master server and type:

```
oldmaster# /usr/etc/yp/ypxfr -h newmaster jokes.bypunchline
```

Now it is safe to run `yppush`. The remaining slave servers still believe that the old master is the current master. They will attempt to get the current version of the map from the old master. When they do so, they will get the new map, which names the new master as the current master.

If this method fails, you can try this cumbersome but sure-fire option. Log in as superuser on each NIS server and execute the `ypxfr` command shown above. This will certainly work, but you should consider it the worst case solution.

Using NIS in conjunction with DNS

Host lookups may be configured in one of the following ways:

- Through NIS exclusively, through the `hosts.byname` and `hosts.byaddr` maps as outlined above; this is effective only within a given domain.

- The Domain Name Service (DNS) can be used for all hostname and address lookups if NIS is not running, or
- NIS can be used to look up hostnames and addresses initially, followed by DNS lookup if the NIS lookup fails.

To configure hostname and address lookups to occur exclusively through DNS, please refer to the chapter *Administering Domain Name Service*.

To configure hostname and address lookup to occur through NIS and then through DNS you should follow the following steps:

1. The two maps `hosts.byname` and `hosts.byaddr` must have the `YP_INTERDOMAIN` key set in them; to obtain this you must edit the file `/var/yp/Makefile` and modify the lines (at the top of the file) that say

```
#B=-b
B=
```

to say

```
B=-b
#B=
```

Thus `makedbm` will be started with the `-b` flag when making the maps, and the `YP_INTERDOMAIN` key will be inserted in them.

2. All NIS servers should have an `/etc/resolv.conf` file that points to a valid nameserver (see *Administering Domain Name Service* for instructions on how to write this file).

Problems in Heterogeneous NIS Domains

The above assumes that both master and slave servers in the NIS domain are running Release 4.x of the SunOS operating system. If that is not the case, problems may arise. The following is a list of possible problems, and some possible workarounds.

- If an NIS server, either master or slave, is running release 3.x, interdomain operations are not recognized by its `ypserv`.

Workaround: Start `ypserv` on that server with the (undocumented) option `-i`.

- If the master server is running 3.x and the slave server is running 4.x, the `YP_INTERDOMAIN` key will not be present in the maps.

Workaround: Start `ypxfr` with the `-b` option on the slave, if the slave is running 4.0.3 or a later release. If the slave is running 4.0 or 4.0.1, upgrade it to 4.0.3 or a later release.

- If the master server is running 4.x prior to 4.0.3 and the slave server is running 4.x, the `YP_INTERDOMAIN` key will be stripped from the maps when a transfer is done.

Workaround: Start `ypxfr` with the `-b` option on the slave, if the slave is running 4.0.3 or a later release. If the slave is running 4.0 or 4.0.1, upgrade to 4.0.3 or a later release.

The following table summarizes the above information. The notation “4.0.3-” means “release of the SunOS operating system earlier than 4.0.3” — that is, 4.0 or 4.0.1”. The notation “4.0.3+” means “release 4.0.3 of the SunOS operating system or later”. The command `makedbm -b` is a reference to the `B` variable in the Makefile, as explained in the previous section.

Table 16-2 *NIS/DNS in Heterogeneous NIS Domains*

slave	master		
	3.x	4.0.3-	4.0.3+
3.x	on master: <code>ypserv -i</code> on slave: <code>ypserv -i</code>	on master: <code>makedbm -b</code> on slave: <code>ypserv -i</code>	on master: <code>makedbm -b</code> on slave: <code>ypserv -i</code>
4.0.3-	upgrade slave	upgrade slave	upgrade slave
4.0.3+	<code>ypserv -i</code> on master <code>ypxfr -b</code> on slave	<code>makedbm -b</code> on master <code>ypxfr -b</code> on slave	<code>makedbm -b</code> on master <code>ypxfr</code> on slave

16.7. Summary of NIS Related Commands

In addition to maps, NIS service also includes specialized daemons, system programs, and commands, which are summarized below. Some of them have been introduced in detail before, some of them have not. In all cases, refer to their man pages.

<code>ypserv</code>	Looks up requested information in a map. <code>ypserv</code> is a daemon that runs on NIS servers with a complete set of maps. At least one <code>ypserv</code> daemon must be present on the network for NIS service to function.
<code>ypbind</code>	Provides NIS server binding information to clients. It provides binding by finding a <code>ypserv</code> process that serves maps within the domain of the requesting client. <code>ypbind</code> must run on all servers and clients.
<code>ypinit</code>	Automatically creates maps for an NIS server from the input files. It also constructs the initial maps that are not built from files, such as <code>ypservers</code> . Use <code>ypinit</code> to set up the master NIS server and the slave NIS servers for the first time.
<code>make</code>	Updates NIS maps by reading the Makefile in <code>/var/yp</code> . You can use <code>make</code> to update all maps based on the input files or to update individual maps. The man page <code>ypmake(8)</code> describes <code>make</code> 's functionality for NIS.
<code>makedbm</code>	<code>makedbm</code> takes an input file and converts it into <code>dbm .dir</code> and <code>.pag</code> files—valid <code>dbm</code> files that NIS can use as maps. You can

also use `makedbm -u` to “disassemble” a map, so that you can see the key-value pairs that comprise it.

<code>ypxfr</code>	Moves an NIS map from one server to another, using NIS itself as the transport medium. You can run <code>ypxfr</code> interactively, or periodically from a <code>crontab</code> file. It is also called by <code>ypserv</code> to initiate a transfer.
<code>ypxfrd</code>	Provides faster map transfers for 4.1 slave servers. It is run only on the master server.
<code>yppush</code>	Copies a new version of an NIS map from the NIS master server to its slaves. You run it on the master NIS server.
<code>ypset</code>	Tells a <code>ypbind</code> process to bind to a named NIS server. This is not for casual use and its use is discouraged because of security implications. See the man page for <code>ypset(8)</code> and <code>ypserv(8)</code> in the <i>SunOS Reference Manual</i> , particularly in reference to the <code>-ypset</code> and <code>-ypsetme</code> options to the latter.
<code>yppoll</code>	Tells which version of an NIS map is running on a server that you specify. It also lists the master server for the map.
<code>ypcat</code>	Displays the contents of an NIS map.
<code>ypmatch</code>	Prints the value for one or more specified keys in an NIS map. You cannot specify which NIS server’s version of the map you are seeing.
<code>ypwhich</code>	Shows which NIS server a client is using at the moment for NIS services, or, if invoked with the <code>-m mapname</code> option, which NIS server is master of each of the maps. If only <code>-m</code> is used, it displays the names of all the maps available and their respective master servers.
<code>ypupdated</code>	Facilitates the updating of NIS information.

16.8. Fixing NIS Problems

This section explains how to clear problems encountered on networks running NIS. It has two parts, covering problems seen on an NIS client and those seen on an NIS server.

Debugging an NIS Client

Before trying to debug an NIS client, review the first part of the chapter, which explains the NIS environment. Then look for the subheading in this section that best describes your problem.

Hanging Commands on the Client

The most common problem of NIS clients is for a command to hang and generate console messages such as:

```
NIS: server not responding for domain <domainname>. Still trying
```

Sometimes many commands begin to hang, even though the system as a whole seems fine and you can run new commands.

The message above indicates that `ypbind` on the local machine is unable to communicate with `ypserv` in the domain `domainname`. This happens when a machine running `ypserv` has crashed and there is not other NIS server available on your subnet (otherwise `ypbind` would broadcast for another server and obtain it.) It may also occur if the network or NIS server is so overloaded that `ypserv` cannot get a response back to the client's `ypbind` within the timeout period.

Under these circumstances, every client on the network will experience the same or similar problems. The condition is temporary in most cases. The messages will usually go away when the NIS server reboots and restarts `ypserv`, or when the load on the NIS servers or network itself decreases.

However, commands may hang and require direct action to clear them. The following list describes the causes of such problems and gives suggestions for fixing them:

- The NIS client has not set, or has incorrectly set, `domainname` on the machine. Clients must use a domain name that the NIS servers know.

On the client type `domainname` to see which domain name is set. Compare that with the actual domain name in `/var/yp` on the NIS master server. If a machine's `domainname` is not the same as the server's, the machine's `domainname` entry in `/etc/defaultdomain` is incorrect. Log in as `superuser` and correct the client's `defaultdomain` file. This assures domain name is correct every time the machine boots. Reboot the machine.

- If your domain name is correct and commands still hang, make sure the subnet has at least one NIS server machine. Some networks consist of two or more subnets joined by routers. A client can automatically bind only to a `ypserv` process on its subnet, not on another accessible network or subnet. At least one NIS server for a client's domain must be running on the client's subnet. Two or more NIS servers improve availability and response characteristics for NIS services.

In extreme cases where local server binding is not possible, use of the `ypset` command may temporarily allow binding to another server, if available, on another network or subnet. However, if `ypbind` has not been started with either of the `-ypset` or `-ypsetme` options, this kind of binding is not possible. Notice that for security reasons the use of `-ypset` should be limited only to debugging purposes under controlled circumstances, and that the use of `-ypsetme` can also result in serious security breaches. If you must start `ypbind` with the `-ypsetme` option, once you have fixed the problem you should kill `ypbind` and restart it again without the option.

- If your local network has an NIS server and commands still hang, make sure the server is up and running. Check other machines on your local network. If several clients also have problems, suspect a server problem. Try to find a client machine behaving normally, and type the `ypwhich` command on it. If `ypwhich` does not respond, kill it and go to a terminal on the NIS server.

Type:

```
# ps ax | grep yp
```

Look for `ypserv` and `ypbind` processes. If the server's `ypbind` daemon is not running, start it up by typing:

```
# ypbind
```

If a `ypserv` process is running, type

```
# ypwhich
```

on the NIS server. If `ypwhich` does not respond, `ypserv` has probably hung, and you should restart it. While logged in as superuser, type the following:

```
# ps axu | grep ypserv
# kill -9 [pid # from ps]
# ypserv
```

If `ps` shows no `ypserv` process running, start one up.

NIS Service is Unavailable

When most machines on the network appear to be fine, but one client cannot receive NIS service, that client may experience many different symptoms. For example, some commands appear to operate correctly while others terminate with an error message about the unavailability of NIS. Other commands limp along in a backup-strategy mode particular to the program involved. Still other commands or daemons crash with obscure messages or no message at all. Here are messages a client in this situation may receive:

```
samba% ypcat myfile
ypcat: can't bind to NIS server for domain <domainname>.
Reason: can't communicate with ypbind.
```

```
% /usr/etc/yp/ypoll myfile
Sorry, I can't make use of the NIS. I give up.
```

On the problem client, run `ls -l` on a directory containing files owned by many users, including some not in the client's `/etc/passwd` file—for example `/usr`. If the resulting display lists file owners not in the local `/etc/passwd` as numbers, rather than names, this also means that NIS service is not working.

These symptoms usually indicate that the client's `ypbind` process is not running. Run `ps ax` and check for `ypbind`. If you do not find it, log in as superuser and start it as follows:

```
# ypbind
```


ypbind Crashes

NIS problems should disappear.

If `ypbind` crashes almost immediately each time it is started, look for a problem in some other part of the system. Check for the presence of the `portmap` daemon by typing:

```
% ps ax | grep portmap
```

If it is not running, reboot.

If `portmap` itself will not stay up or behaves strangely, look for more fundamental problems. Check the network software in the ways suggested in the section on Ethernet debugging in the chapter *The SunOS Network Environment*.

You may be able to communicate with `portmap` on the problematic client from a machine operating normally. From the functioning machine, type:

```
% rpcinfo -p client
```

If `portmap` on the problematic machine is fine, `rpcinfo` produces the following output:

```
program vers proto  port
100007    2    tcp    1024  ypbind
100007    2    udp    1028  ypbind
100007    1    tcp    1024  ypbind
100007    1    udp    1028  ypbind
100021    1    tcp    1026  nlockmgr
100024    1    udp    1052  status
.
.
.
```

Your machine will have different port numbers. The four entries for the `ypbind` process are:

```
100007    2    tcp    1024  ypbind
100007    2    udp    1028  ypbind
100007    1    tcp    1024  ypbind
100007    1    udp    1028  ypbind
```

If they are not displayed, `ypbind` has been unable to register its services. Reboot the machine and run `rpcinfo` again. If the `ypbind` processes are there and they change each time you try to restart `/usr/etc/ypbind`, reboot the system, even if the `portmap` daemon is running. If the situation persists after reboot, call Sun customer support for help.

ypwhich Displays are Inconsistent

When you use `ypwhich` several times on the same client, the resulting display varies because the NIS server changes. This is normal. The binding of NIS client to NIS server changes over time when the network or the NIS servers are busy. Whenever possible, the network stabilizes at a point where all clients get acceptable response time from the NIS servers. As long as your client machine gets NIS service, it does not matter where the service comes from. For example, an NIS server machine may get its own NIS services from another NIS server on the network.

Debugging an NIS Server

Before trying to debug your NIS server, read the beginning part of this chapter about the NIS environment. Then look in this subsection for the heading that most closely describes the server's problem.

Servers Have Different Versions of an NIS Map

Because NIS propagates maps among servers, occasionally you may find different versions of the same map on various NIS servers on the network. This version discrepancy is normal if transient, but abnormal otherwise.

Most commonly, normal map propagation is prevented if it occurs when an NIS server or router between NIS servers is down. When all NIS servers and the routers between them are running, `ypxfr` should succeed.

If a particular slave server has problems updating maps, log in to that server and run `ypxfr` interactively. If `ypxfr` fails, it will tell you why it failed, and you can fix the problem. If `ypxfr` succeeds, but you suspect it has occasionally failed, create a log file to enable logging of messages. As superuser type:

```
ypslave# cd /var/yp
ypslave# touch ypxfr.log
```

This saves all output from `ypxfr`. The output resembles the output `ypxfr` displays when run interactively, but each line in the log file is timestamped. (You may see unusual orderings in the timestamps. That is okay—the timestamp tells you when `ypxfr` started to run. If copies of `ypxfr` ran simultaneously but their work took differing amounts of time, they may actually write their summary status line to the log files in an order different from that which they were invoked.) Any pattern of intermittent failure shows up in the log. When you have fixed the problem, turn off logging by removing the log file. If you forget to remove it, it will grow without limit.

While still logged in to the problem NIS slave server, inspect `root`'s `crontab` file on that machine, and check the `ypxfr*` shell scripts it invokes. Typos in these files cause propagation problems, as do failures to refer to a shell script within `/var/spool/cron/crontabs/root`, or failures to refer to a map within any shell script.

Also, make sure that the NIS slave server is in the map `ypservers` within the domain. If it is not, it still operates perfectly as a server, but `yppush` will not tell it when a new copy of a map exists.

If the NIS slave server's problem is not obvious, you can work around it while you debug using `rcp` or `tftp` to copy a recent version of the inconsistent map

from any healthy NIS server. Be sure not to do this remote copy as root, but you can probably do it while logged in as daemon. For instance, here is how you might transfer the map “busted:”

```
yplslave# chmod go+w /var/yp/mydomain
yplslave# su daemon
$ rcp ypmaster:/var/yp/mydomain/busted.* /var/yp/mydomain
$ ^D
yplslave# chown root /var/yp/mydomain/busted.*
yplslave# chmod go-w /var/yp/mydomain
```

Here the * character has been escaped in the command line, so that it will be expanded on *ypmaster*, instead of locally on *yplslave*. Notice that the map files should be owned by root, so you must change ownership of them after the transfer.

ypserv Crashes

When the *ypserv* process crashes almost immediately, and does not stay up even with repeated activations, the debug process is virtually identical to that previously described in the subsection “*ypbind* Crashes.” Check for the existence of the *portmap* daemon as follows:

```
ypserver% ps ax | grep portmap
```

Reboot the server if you do not find the daemon. If it is there, type:

```
% rpcinfo -p yp_server
```

and look for output such as:

```

program vers proto  port
100004    2   udp   1027  ypserv
100004    2   tcp   1024  ypserv
100004    1   udp   1027  ypserv
100004    1   tcp   1024  ypserv
100007    2   tcp   1025  ypbind
100007    2   udp   1035  ypbind
100007    1   tcp   1025  ypbind
100007    1   udp   1035  ypbind
100009    1   udp   1023  yppasswdd
100003    2   udp   2049  nfs
100024    1   udp   1074  status
100024    1   tcp   1031  status
100021    1   tcp   1032  nlockmgr
100021    1   udp   1079  nlockmgr
100020    1   udp   1082  llockmgr
100020    1   tcp   1033  llockmgr
100021    2   tcp   1034  nlockmgr
100012    1   udp   1104  sprayd
100011    1   udp   1106  rquotad
100005    1   udp   1108  mountd
100008    1   udp   1110  walld
100002    1   udp   1112  rusersd
100002    2   udp   1112  rusersd
100001    1   udp   1115  rstatd
100001    2   udp   1115  rstatd
100001    3   udp   1115  rstatd

```

Your machine will have different port numbers. The four entries representing the ypserv process are:

```

100004    2   udp   1027  ypserv
100004    2   tcp   1024  ypserv
100004    1   udp   1027  ypserv
100004    1   tcp   1024  ypserv

```

If they are not present, ypserv has been unable to register its services. Reboot the machine. If the ypserv processes are there, and they change each time you try to restart /usr/etc/ypserv, reboot the machine. If the situation persists after reboot, call Sun for assistance.

16.9. Turning Off NIS Services

If ypserv on the master is disabled, you can no longer update any the NIS maps. If you choose to turn off NIS on a network currently running it, you can disable it by simply renaming the /usr/etc/ypbind file to /usr/etc/ypbind.orig. SunInstall does this automatically if you tell it you do not want to run NIS. Type the following:

```

% cd /etc
% mv /usr/etc/ypbind /usr/etc/ypbind.orig

```

To disable NIS on a particular NIS slave or master, type the following on the server in question:

```
% mv /usr/etc/ypserv /usr/etc/ypserv.orig.
```

Again, SunInstall does this automatically if you do not select NIS.

Refer back to the chapter on *The Sun Network File System Service*, for information about the files you need to maintain for a server should you decide to disable NIS.

16.10. Default NIS Maps

The following table describes the default NIS maps, information they contain, and whether SunOS consults the corresponding administrative files when NIS is running.

Table 16-3 *NIS Map Descriptions*

Map Name	Corresponding Admin File?	Description
bootparams	bootparams	Contains pathnames of files clients need during booting: root, swap, possibly others. With NIS, bootparamd does not consult /etc/bootparams after map is created.
ethers.byaddr	ethers	Contains machine names and Ethernet addresses. The Ethernet address is the key in the map. /etc/ethers is not consulted after map is created.
ethers.byname	ethers	Same as ethers.byaddr, except key is machine name instead of Ethernet address.
group.bygid	group	Contains group security information with group ID as key. With NIS, local /etc/group is consulted first, then map.
group.byname	group	Contains group security information with group name as key. With NIS, local /etc/group is consulted first, then map.
hosts.byaddr	hosts	Contains host name, and IP address, with IP address as key. With NIS, the map is always used except at boot time, when /etc/hosts is consulted.
hosts.byname	hosts	Contains hostname and IP address, with hostname as key. With NIS, map is always used, except at boot time, when it /etc/hosts is consulted.
mail.aliases	aliases	Contains aliases and mail addresses, with aliases as key. With NIS, local /etc/aliases is consulted first, then map.

Table 16-3 NIS Map Descriptions—Continued

Map Name	Corresponding Admin File?	Description
mail.byaddr	aliases	Contains mail address and alias, with mail address as key. With NIS, local /etc/aliases is consulted first, then map.
netgroup.byhost	netgroup	Contains group name, user name and host name, with host name as key. /etc/netgroup is never consulted.
netgroup.byuser	netgroup	Same as netgroup.byhost, except that key is user name.
netgroup	netgroup	Same as netgroup.byhost, except that key is group name.
netid.byname	passwd, hosts, group	Used for UNIX-style authentication. Contains machine name and mail address (including domain name). If there is a netid file available it is consulted in addition to the data available through the other files.
netmasks.byaddr	netmasks	Contains network mask to be used with IP subnetting, with mask as key. /etc/netmasks is never consulted after the map is created.
networks.byaddr	networks	Contains names of networks known to your system and their IP addresses, with the address as the key. /etc/networks is never consulted after map is created.
networks.byname	networks	Same as networks.byaddr, except key is name of network.
passwd.byname	password	Contains password information with user name as key. With NIS, local /etc/passwd is consulted first, then map.
passwd.byuid	password	Same as passwd.byname, except that key is user ID.
protocols.byname	protocols	Contains network protocols known to your network, with protocol name as key. /etc/protocols is never consulted after map is created.
protocols.bynumber	protocols	Same as protocols.byname, except that key is protocol number.
publickey.byname	publickey	Contains public and secret keys. /etc/publickey is never consulted after map is created.
rpc.bynumber	rpc	Contains program number and name of RPCs known to your system. Key is RPC program number. /etc/rpc is never consulted after map is created.

Table 16-3 *NIS Map Descriptions—Continued*

Map Name	Corresponding Admin File?	Description
services.byname	services	Lists Internet services known to your network. Key is service name. /etc/services is never consulted after map is created.
ypservers	None	Lists NIS servers known to your network.

Administering Domain Name Service

17.1. What is DNS?

Domain Name Service (DNS) is an Applications layer protocol that is part of the standard TCP/IP protocol suite.

DNS is a *naming* service (i.e. a service that associates information with objects). Its basic function is to obtain and provide information about hosts on a network by querying and answering queries. It differs from other naming services in that:

- The information it uses to answer queries is not centralized in a single file or machine.
- It can resolve queries across domains.
- It initially accesses the information only as needed.
- It maintains an updatable cache of the information obtained, thus eliminating the need to obtain the same information repeatedly.

DNS is also a networking concept and the implementation of that concept in libraries and services. This chapter explains how to set up DNS, and includes the following topics:

- DNS domain organization
- DNS name space
- Servers and clients
- Programs and files composing DNS
- Procedures for setting up DNS
- Procedures for debugging DNS
- Responsibilities of domain administrators and technical/zone contacts

17.2. The DNS Domain Hierarchy

Domain Name Service performs naming between hosts *within* your local administrative domain and *across* domain boundaries. It is distributed among a set of servers, each of which implements DNS by running a daemon called `in.named`. These servers are commonly referred to as *name servers*.

Note: The `in.named` daemon is also sometimes called the Berkeley Internet Name Domain service, or *BIND*, because it was developed at University of California at Berkeley. Throughout this chapter and in the literature it is also referred to as the `named` daemon, without the prefix.

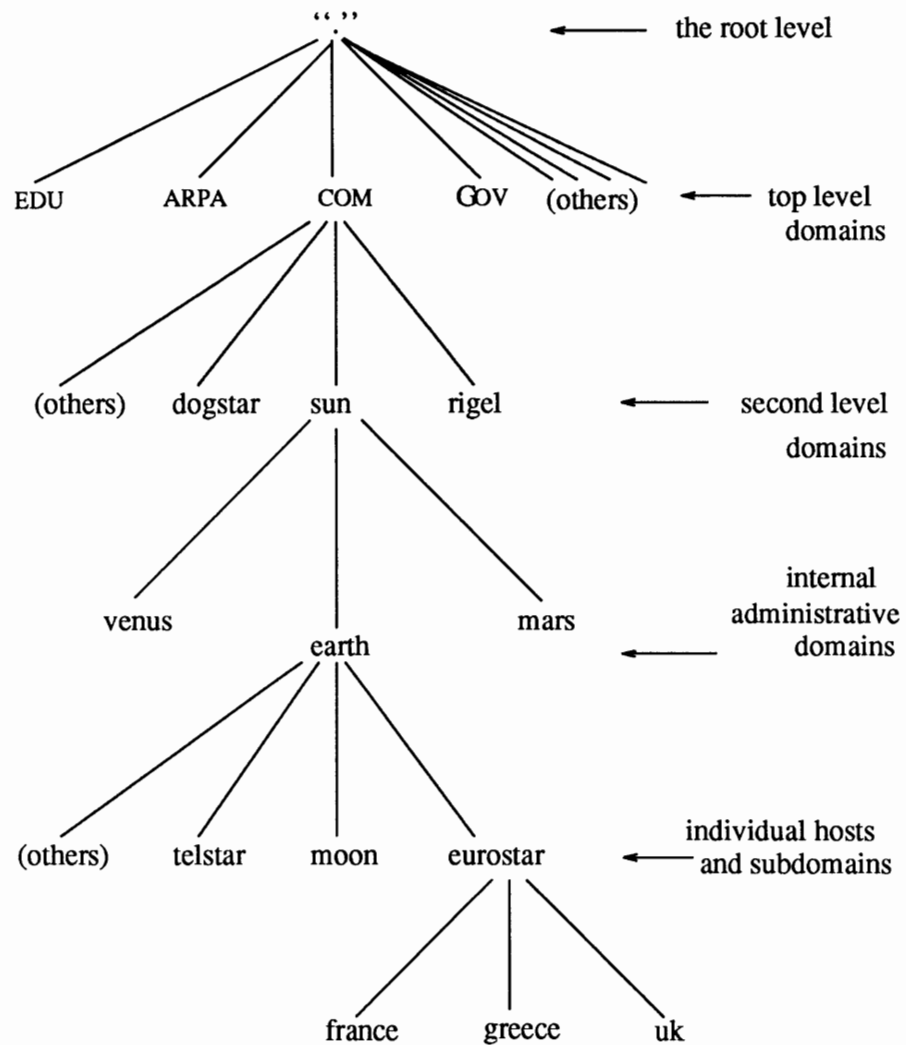
On the client's side, the service is implemented through the *resolver*. The resolver is neither a daemon nor a particular program; rather it is part (usually in a library) of those applications that need to know machines' locations. Its function is to resolve user queries; for that it needs to know the address of at least one name server. The name server then can return either the desired answer or a referral to another server.

A name server can maintain two kinds of data. The first kind, called "authoritative", refers to the zones maintained by the server in question (see below, *Name Space and Administrative Zones*, for a discussion of zones). The server updates this data periodically. The second kind of data maintained by the server is cached data, obtained through the local resolver. This data is not authoritative and it may be incomplete or out of date, but it speeds up the retrieval process for non local data. There is a timeout mechanism to ensure that cached data is eventually discarded and updated.

Note: The names of second- and lower-level domains in this and all other examples in this chapter are purely imaginary, and should not be construed as implying the existence of those domains.

DNS names have a hierarchical organization, consisting of domains nested within one another like SunOS directories. Names are written from bottom to top, with dots separating the levels. Before you actually set up DNS for your organization, you should understand this hierarchy. It not only reflects your place in the overall domain structure, but also how you name domains and hosts—an important aspect of name service administration. The figure on the next page uses the organization of the Internet as an example of DNS hierarchy.

Figure 17-1 A Typical Internet Domain Hierarchy



The Root Level Domain

The root of the tree, the top of the entire Internet, is currently maintained by the NIC (Network Information Center). This is why you must contact the NIC in order to join the Internet. Organizations in charge of other public networks, such as BITNET and CSNET administer this area for their networks.

At the root level, the NIC administers *root domain name servers* that maintain information about name servers at the next lower level. Your organization's name servers will forward requests they cannot answer to these root servers. For example, the Internet root name server TERP.UMD.EDU is one of several serving the Eastern United States; NIC.DDN.MIL (formerly named SRI-NIC.ARPA) is one of several servicing the Western U.S. (The later subsection discussing the named .ca file further explains root name servers.)

Top Level Domains

The root of the tree branches into *top level domains*. Currently, some Internet top level domains are `EDU`, `ARPA`, `COM`, `GOV`, `MIL`, `NET`, `ORG` and `US`, as shown in the previous figure. When you register with the NIC, it assigns your network to one of these domains, depending on your organization's function. For example, the NIC currently assigns educational institutions to the `EDU` domain and business institutions to the `COM` domain. The NIC also maintains DNS for these top level domains. Note that `US` and other country names are also valid top level domains.

Second Level Domains

Each top level domain branches into second level domains, such as `dogstar`, `sun`, and `rigel` in the example above. *Domain Administrators* working for the member organizations of the domain manage the name servers at this level. Depending upon your site's size and requirements, you may have domain administrator responsibilities. (A later section, *Administering DNS for Your Domain*, describes these.)

Local Administrative Domains

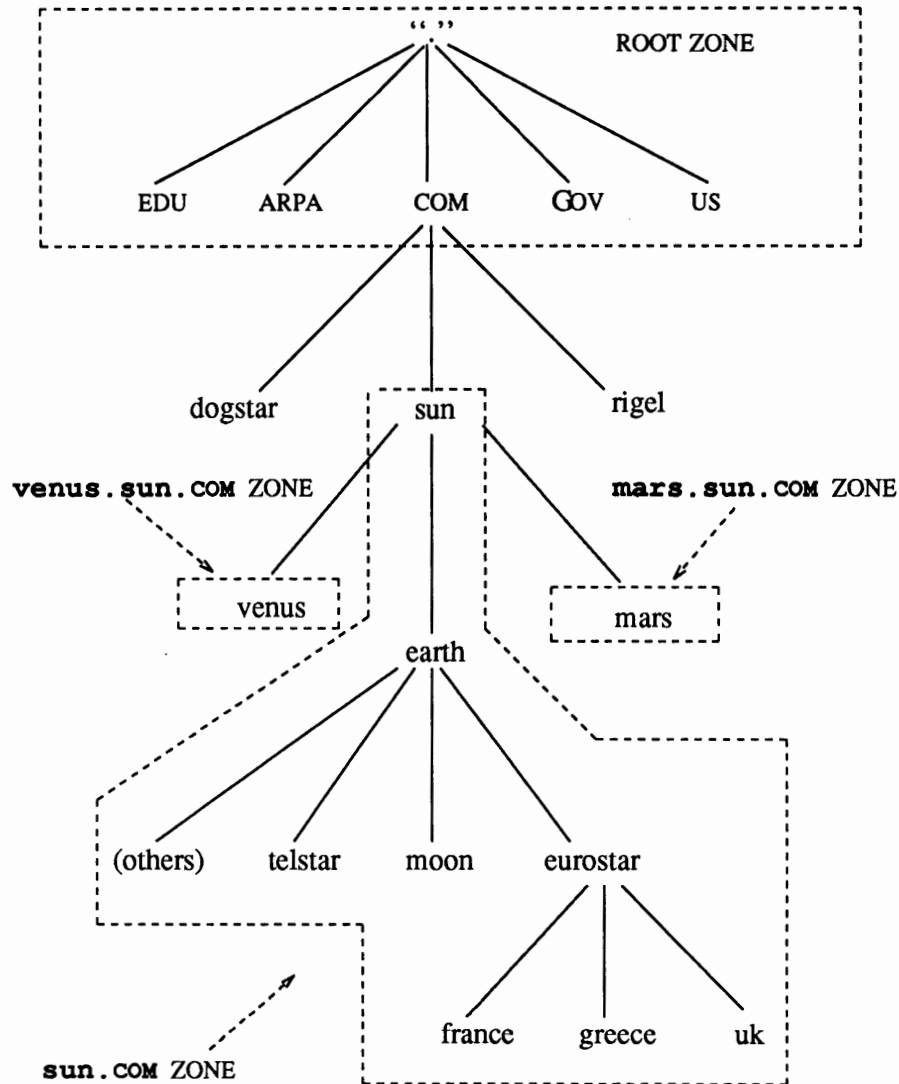
Within each second level are local administrative domains. These are the domains you administer for your organization. A second level domain can be as small as one host, or large enough to include many hosts and additional name servers. It may also have other administrative domains nested within it. In the previous figure, second level domain `sun.com` has three internal administrative domains: `venus`, `earth`, and `mars`. Domain `earth` includes individual hosts such as `moon`, plus a name server `eurostar` serving additional hosts.

17.3. Name Space and Administrative Zones

The Domain Name Service introduces the concept of *zones*. A zone is a hierarchical community of hosts, administered by a single authority and served by a set of name servers. This community can include individual hosts, plus every name server and its clients (a subzone) nested beneath the authoritative set of name servers for the zone. Zones usually represent administrative boundaries, such as your local administrative domain. However, a zone is not limited to just one administrative domain. A zone is a domain, plus all domains under it, except those delegated.

The next figure shows how some fictional zones might be delegated. The illustration shows four zones: the root zone, served by the Internet root domain name servers, and three zones served by name servers at domain `sun.com`. The dotted lines delimit the areas making up each zone. The zone name appears in boldface.

Figure 17-2 Zones in the Internet Name Space



How Name Space Relates to Host and Domain Names

As you read through this manual, you probably noticed many instances where you must specify a host's name as a fully-qualified domain name. This name reflects the host or current domain's position in the name space; it comprises the ordered list of labels from the root to the host or current domain.

Earlier in this guide, you learned that fully-qualified domain names are similar to SunOS full pathnames with two significant differences: Domain names are listed from right to left and are delimited by periods rather than slashes. Here is how this ordering relates to the name space shown in the previous figure.

Consider the following sample domain names:

```
COM. rigel.COM. venus.sun eurostar.earth.sun.COM.
```

When a period appears as the rightmost character of the domain name, it

represents the null label of the root. If you do not specify this trailing period, DNS assumes that the domain name is “relative” to a known origin, as in the domain name `venus.sun` above.

You use the same rule of thumb for specifying the fully qualified domain name of a host in your local network. For example, consider the machine named `uk` in the subdomain `eurostar`. Its fully qualified domain name is:

```
uk.eurostar.earth.sun.com.
```

How Name Space Relates to Zones

Zones take their names from the label of the domain at the top of the zone hierarchy. In the last figure, there are four domains within the dotted lines:

```
sun.com  venus.sun.com  mars.sun.com  . (the root zone)
```

Zone `sun.com` takes its name from the label of the second level domain `sun.com`. However, zone `sun.com` does not have the same administrative authority as the domain of the same name. Zone `sun.com` consists only of:

- domain `sun`
- local administrative domain `earth`
- subdomain `eurostar`.

The zone `sun.com` does not include domains `venus` and `mars`. They have their own separately administered zones: `venus.sun.com` and `mars.sun.com`, respectively. However, `venus` and `mars` are part of the `sun.com` domain hierarchy.

Name Space and the IN-ADDR.ARPA Domain

The domain hierarchy and name space described so far keeps track of information by host name. This enables the `in.named` daemon to perform name to address mapping.

In addition, there is a special domain the DNS recognizes, called `IN-ADDR.ARPA`, which facilitates address to name mapping. `IN-ADDR.ARPA` contains the same information as the hosts name space, but it is expressed in terms of IP addresses.

A name in the `IN-ADDR.ARPA` domain has four labels preceding it, corresponding to the four octets of an IP address. This host address is listed from right to left. For example, a host whose IP address is `128.32.0.4` has the `IN-ADDR.ARPA` domain name

```
4.0.32.128.IN-ADDR.ARPA.
```

Therefore, if `in.named` knows the IP address of a host, it can find the host's fully qualified domain name by consulting a file representing the `IN-ADDR.ARPA` domain. (This file, sometimes called `hosts.rev`, is explained in a later section.)

DNS Servers and Clients

There are two sides to DNS: name servers running the `in.named` daemon and clients running the resolver. A name server running `in.named` can also run the resolver. Therefore there can be two kinds of clients:

- Client-only
- Server/client

You implement name service for a zone (see above, *Name Space and Administrative Zones*) on a set of servers. This set may include the following:

- Master servers (primary and secondary)
- Caching-only server
- Forwarding server

These are explained below.

Clients

A client-only machine does not run the `in.named` daemon; instead, it consults the file `/etc/resolv.conf`, which provides a list of possible name serving machines to which queries should be directed.

A name server/client is a machine that uses the domain name service provided by `in.named` in order to resolve a user's queries. However, were the daemon to die or hang, that machine might be able to solve queries through its resolver. Thus, a machine running the `named` daemon does not strictly have to have an `/etc/resolv.conf` file, but it is recommended that it does.

NIS servers can be made into resolver clients by using the `-b` option to `makedbm` on the master server for certain maps (for more details, please consult the chapter *The Network Information Service*.) Each NIS server (master as well as slaves) should then either run `in.named`, have an `/etc/resolv.conf` file, or both.

Master Servers

Each zone must have at least two *master* name servers that maintain all the data corresponding to the zone, making them the authority for that zone. In other words, the data corresponding to any given zone must be available on at least two servers. You should designate one name server as the primary master server and at least one additional server as the secondary master server to serve as a backup if the primary is unavailable or overloaded.

The *primary* master server is the name server where you make changes for the zone. This server loads the master copy of its data from disk when it starts up `in.named`. The primary server may also delegate authority to other servers in its zone, as well as to servers outside of it.

The *secondary* master server is a name server that maintains a copy of the data for the zone. The primary server sends its data and delegates its authority to the secondary server. When the secondary server boots `in.named`, it requests all the data for the given zone from the primary. The secondary server then periodically checks with the primary to see if it needs to update its data.

A server may function as a master for multiple zones: as a primary for some zones and as a secondary for others.

Caching and Caching-Only Servers

All name servers are caching servers. This means that the name server caches received information until the data expires. (The expiration process is regulated by the “time to live” field attached to the data when it is received from another server; see below, *Standard Resource Record Format*.)

Additionally, you can set up a *caching only server* that is not authoritative for any zone. This server handles queries and asks other name servers who have the authority for the information needed. But the caching only server does not maintain any authoritative data itself.

Forwarding Servers

You can optionally set up servers that handle requests by forwarding them to other servers. There may be one or more servers in the forwarding list, and they are tried in turn until the list is exhausted. A forwarding server can also be forced to refrain from resolving queries locally by designating it as a slave.

A typical scenario in which you may wish to use this scheme is if you have a large machine connected to the Internet or some other network and a series of smaller machines or workstations with no connection to the outside world. In order to give the workstations the appearance of access to the Internet or some other network, you would set up the small machines as forwarding slaves of the larger one, and their requests would be forwarded to it, which in turn would interact with other servers to resolve the query and return the answer.

An added advantage is that the machine to which the queries are forwarded would be able to build a very rich cache of data compared to the cache a typical workstation would build, thereby reducing the number of queries from the site to the rest of the net.

17.4. Name Server Files

The domain name server uses several files to load its database. At the resolver level, it needs a file (called `/etc/resolv.conf`) listing the addresses of the servers where it can obtain the information needed. Whenever the resolver has to find the address of a host (or the name corresponding to an address) it builds a query packet and sends it to the name servers it knows of (from `/etc/resolv.conf`). The servers either answer the query locally or use the services of other servers and return the answer to the resolver.

At the `named` level, it first needs a boot file (generally called `/etc/named.boot`) that establishes whether the server is a primary, a secondary, a cache-only or a forwarding name server. Furthermore, it establishes the location of other servers and files where information can be obtained.

These files should be in place before you first call up the `named` daemon. When first called, the daemon reads all the files in question and caches the information; as it answers query packets through the services of other servers it increases its cache. Killing the daemon for any reason destroys the cache.

What follows is a detailed discussion of the files used by the resolver and by the `named` daemon, and their formats. The following applies for all the files:

- A semicolon (;) is used to start a comment; whatever follows it in the line is ignored by `named`.

- Parentheses are used to extend a line over several lines for readability reasons. The resolver configuration file and the named boot file have their own syntax; all the other files follow the Resource Record standard syntax, which is examined in detail later on.

Resolver Configuration File

This file is read by the resolver to find out the name of the local domain and the location of name servers. The following is a sample `resolv.conf` file:

```
; Sample resolv.conf file
domain Podunk.Edu
nameserver 128.32.0.4
nameserver 128.32.0.10
```

This sets the local domain to `Podunk.Edu` and instructs the resolver routines to query the listed name servers for information.

This is the only file used by the resolver.

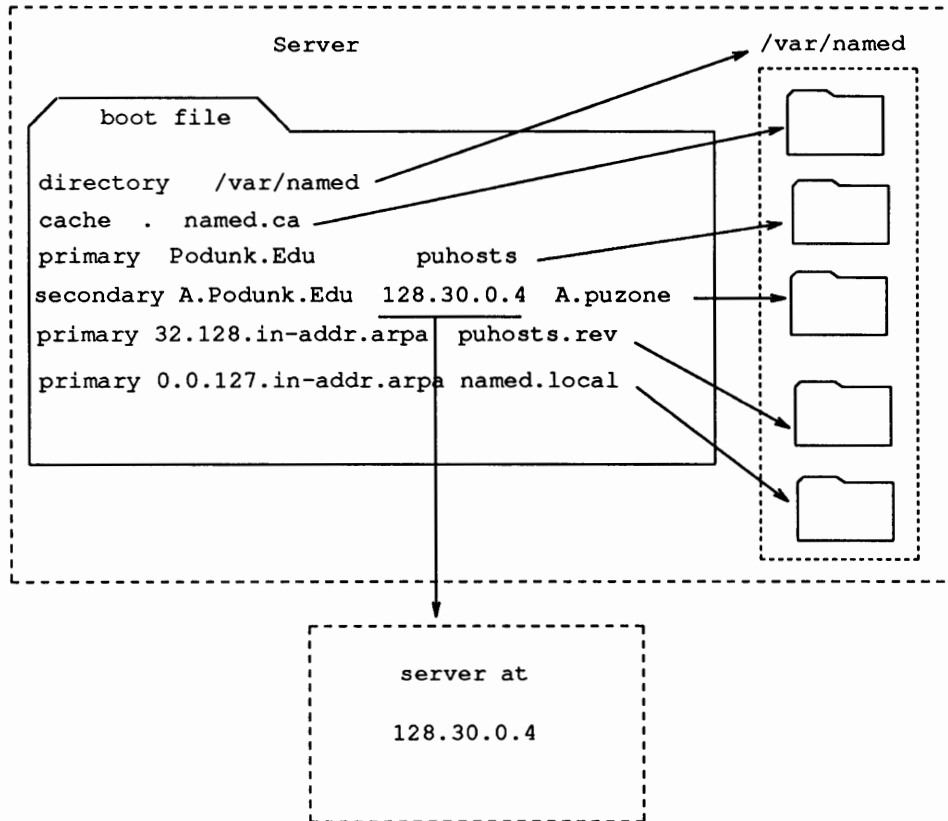
Boot File

This file is first read when `named` is started from the server's `/etc/rc.local` file. The boot file tells the server what type of server it is, which zones it has authority over, and where to get its initial data. Its default location is `/etc/named.boot`. However, you can change its location by specifying another name on the command line when `named` is started (see the man page for `named(8)`).

The contents of a boot file will vary depending on what type of server one is setting up.

The boot file also directs `named` to either other servers or to local data files for a specified domain. Common names for these data files are `named.local`, `hosts` and `host.rev`.

The relationship between the boot file and the data files (or, alternatively, other servers) is best illustrated by the following figure:

Figure 17-3 *Boot file and data files*

All data files use the Standard Resource Record Format discussed later in this section.

Boot File For a Primary Server

The following is a sample boot file for a primary server:

```

;
; Sample named.boot file for Primary Master Name Server
;
; type domain source file or host
;
directory /var/named
cache . named.ca
primary Podunk.Edu puhosts
primary 32.128.in-addr.arpa puhosts.rev
primary 0.0.127.in-addr.arpa named.local

```

Analyzing this file line by line:

```
directory /var/named
```

This line in the boot file designates the directory in which you want the name

server to run. This allows the use of relative path names for the files mentioned in the boot file or, later, with the `$INCLUDE` directive (see below). It is especially useful if you have many files to be maintained and you want to locate them all in one directory dedicated to that purpose. It is also useful for making sure that if named dumps core it will do so in the directory you specify. You are free to choose any directory, although any directory under `/var` would be a good choice, to allow for the growth of the files.

If there is no `directory` line in the boot file all file names listed in it must be absolute pathnames (i.e. must start at the root directory).

```
cache . named.ca
```

A name server needs to know which servers are the authoritative name servers for the root zone. To do this, you have to list the addresses of these higher authorities.

All servers should have the above line in the boot file to find the root name servers. The first field (`cache`) indicates that the server will obtain root servers hints from the indicated file, in this case `named.ca` (located in the directory `/var/named`).

The third field `named.ca` (above) is the name of the file that lists the root servers. The name of this file is generally `named.ca`, although to avoid confusion with the caching process, with which this has very little to do, you may want to call your file something like `named.root` or `boot.root`.

The following is a sample `named.ca` file:

```
;
;Initial cache data for root domain servers.
;
; list of servers...

          99999999 IN NS NIC.DDN.MIL.
          99999999 IN NS A.ISI.EDU.
          99999999 IN NS TERP.UMD.EDU.
          99999999 IN NS C.NYSER.NET.
          99999999 IN NS AOS.BRL.MIL.
          99999999 IN NS GUNTER-ADAM.AF.MIL.

; ...and their addresses
NIC.DDN.MIL.      99999999 IN A 10.0.0.51
NIC.DDN.MIL      99999999 IN A 26.0.0.73
C.NYSER.NET.     99999999 IN A 192.33.4.12
AOS.BRL.MIL.     99999999 IN A 128.20.1.2
AOS.BRL.MIL.     99999999 IN A 192.5.22.82
NS.NASA.GOV.     99999999 IN A 128.102.16.10
TERP.UMD.EDU.    99999999 IN A 10.1.0.17
A.ISI.EDU.       99999999 IN A 26.3.0.103
GUNTER-ADAM.AF.MIL. 99999999 IN A 26.1.0.13
```

This file contains information pertaining to the root authoritative servers for the Internet. This information is correct at the time of this printing, but it may change at any time. It uses the Standard Resource Record Format discussed later in this chapter. It first establishes the names of the root servers and then lists their addresses. `named` cycles through the list of servers until it contacts one of them. It then obtains from it the most current list of root servers, and it uses it to update the file, if need be.

```
primary Podunk.Edu puhosts
```

This line in the boot file designates the server as a primary server for a zone. The first field (`primary`) designates the server as primary for the zone stated in the second field (`Podunk.Edu`). The third field (`puhosts`) is the name of the hosts file from which data is read.

This file contains all the data about the machines in this zone. It does not have to be called `hosts` (in fact, it should not, to avoid confusion with `/etc/hosts`); for instance, in the example above it is called `puhosts`. Its name may contain the suffix `zone`; for instance, it can be called `puhosts.zone`.

The contents of this file follow the standard Resource Record format, explained later in this chapter.

```
primary 32.128.in-addr.arpa puhosts.rev
```

This server is also a primary server for the domains `32.128.in-addr.arpa` (that is, the reverse address domain for `Podunk.Edu`). Data for it is to be found in the **reverse hosts** file, `puhosts.rev` in this example. This file specifies a zone in the `IN-ADDR.ARPA` domain. This is a special domain for allowing address-to-name mapping. As Internet host addresses do not fall within domain boundaries, this special domain was formed to allow inverse mapping. The `IN-ADDR.ARPA` domain has up to four labels preceding it, corresponding to the four octets of an IP address in reverse order. For instance, the IP address `128.32.0.4` corresponds to the domain `4.0.32.128.IN-ADDR.ARPA`.

As with the `hosts` file, this file can be called something else; in the examples in this chapter it is called `puhosts.rev`, `purev.zone` and `hosts.rev`. The contents of this file follow the standard Resource Record format, explained later in this chapter.

```
primary 0.0.127.in-addr.arpa named.local
```

This line establishes that this server is also a primary server for the domain `0.0.127.in-addr.arpa` (that is, the local host loopback), and that the data for it is to be found in the file `named.local`

This file specifies the address for the local loopback interface, or *localhost*, with the network address `127.0.0.1`.

As with all other files, it can be called something else. The contents of this file follow the standard Resource Record format, explained later in this chapter.

Unused lines

The `domain` and `sortlist` lines, which were used in previous versions of `in.named`, are no longer used.

Boot File For a Secondary Server

The following is a sample boot file for a secondary server in the same domain as the above primary server:

```

;
; Sample named.boot file for Secondary Master Name Server
;

; type      domain                source file or host
;
directory /var/named
cache      .                      named.ca
secondary  Podunk.Edu             128.32.0.4 128.32.0.10 128.32.136.22 puhosts.zone
secondary  32.128.in-addr.arpa    128.32.0.4 128.32.0.10 128.32.136.22 purev.zone
primary    0.0.127.in-addr.arpa   named.local

```

In appearance, this file is very similar to the boot file for the primary server; the main difference is to be found in these lines

```

secondary  Podunk.Edu             128.32.0.4 128.32.0.10 128.32.136.22 puhosts.zone
secondary  32.128.in-addr.arpa    128.32.0.4 128.32.0.10 128.32.136.22 purev.zone

```

The word `secondary` establishes that this is a secondary server for the zone listed in the second field, and that it is to get its data from the listed servers (usually the primary server followed by one or more secondary ones); attempts are made in the order in which the servers are listed. If there is a filename after the list of servers (as in the example above), data for the zone will be put into that file as a backup. When the server is started, data is loaded from the backup file, if it exists, and then one of the servers is consulted to check whether the data is still up to date.

This ability to specify multiple secondary addresses allows for great flexibility in backing up a zone.

The interpretation of all other “secondary” lines is similar to the above. Note also that although this is a secondary server for the domains `Podunk.Edu` and `32.128.in-addr.arpa`, this is a primary server for `0.0.127.in-addr.arpa` (the local host).

Boot File for Primary and Secondary Server

A server may act as the primary server for one or more zones, and as the secondary server for one or more zones; it is the mixture of entries in the boot file that determines it.

Boot File for Caching-Only Server

The following is a sample boot file for a caching-only server:

```

;
; Sample named.boot file for Caching Only Name Server
;

; type      domain                source file or host
;
cache                               named.ca

```

You do not need a special line to designate a server as a caching-only server. What denotes a caching-only server is the absence of authority lines like `primary` or `secondary`. A caching-only server does not maintain any authoritative data; it simply handles queries and asks the hosts listed in the file named in the third field for the information needed.

Boot file for Forwarding Server

The following is a sample boot file for a forwarding server:

```

;
; Sample named.boot file for Forwarding Server
;
forwarders 128.32.0.10 128.32.0.4
slave

```

The `forwarders` line indicates that queries that cannot be resolved locally should be directed to the servers listed. The `slave` line indicates that all queries should be directed to the forwarders, and that no attempt should be made to resolve a query locally. Note that you cannot have a `slave` line without having a `forwarders` line.

17.5. Standard Resource Record Format

All data files (for instance, `named.ca`, `named.local`, `hosts`, and `host.rev`) are written in a standard format. The file `named.ca`, presented above, follows this format.

Each line of a data file is a record called a Resource Record (RR) containing the following fields separated by white space:

```
{name} {ttl} class Record Type Record Specific data
```

The order of the fields is always the same; however, the first two are optional (as indicated by the braces), and the contents of the last vary according to the *Record Type* field.

name The first field is the name of the domain that applies to the record. If this field is left blank in a given RR, it defaults to the name of the previous RR.

A domain name in a zone file can be either a fully qualified name, terminated with a dot, or a relative one, in which case the current domain is appended to it.

ttl The second field is an optional time-to-live field. This specifies how long (in seconds) this data will be cached in the database before it is disregarded and new information is requested from a server. By leaving this field blank, the ttl defaults to the minimum time specified in the Start Of Authority resource record discussed below.

If the ttl value is set too low, the server will incur in a lot of repeat requests for data refreshment; if, on the other hand, the ttl value is set too high, changes in the information will not be timely distributed.

Most ttl values should be initially set to between a day (86400) and a week (604800); then, depending on the frequency of actual change of the information, change the appropriate ttl values to reflect that frequency. Also, if you have some ttl entries that have very high values because you know they relate to data that rarely changes, and you know that the data is now about to change, reset the ttl to a low value (3600, one hour, to 86400, one day) until the change takes place, and then change it back to the original high value.

All Resource Records (RR's) with the same name, class and type should have the same ttl.

class The third field is the record class. Only one class is currently in use: IN for the TCP/IP Internet protocol family.

type The fourth field states the type of the resource record. There are many types of RR's; the most commonly used types are discussed below, in the section *Resource Record Types*.

RR data The contents of the data field depend on the type of the particular Resource Record.

Although case is preserved in names and data fields when loaded into the name server, all comparisons and lookups in the name server database are case insensitive. However, this situation may change in the future, and you are advised to be consistent in your use of lower and upper case.

Special characters

The following characters have special meanings:

- . A free standing dot in the name field refers to the current domain.
- @ A free standing @ in the name field denotes the current origin.
- .. Two free standing dots represent the null domain name of the root when used in the name field.
- \X Where X is any character other than a digit (0-9), quotes that character so that its special meaning does not apply. For example, you can use \ to place a dot character in a label.
- \DDD Where each D is a digit, this is the octet corresponding to the decimal number described by DDD. The resulting octet is assumed to be text and is not checked for special meaning.

- () Use parentheses to group data that crosses a line. In effect, line terminations are not recognized within parentheses.
- ; Semicolon starts a comment; the remainder of the line is ignored.
- * An asterisk signifies wildcarding.

Most resource records have the current origin appended to names if they are not terminated by a "." This is useful for appending the current domain name to the data, such as machine names, but may cause problems where you do not want this to happen. A good rule of thumb is to use a fully qualified name ending in a period if the name is not in the domain for which you are creating the data file.

Control entries

The only possible lines that do not conform to the standard RR format in a data file are control entry lines. There are two kinds of control entries:

\$INCLUDE

An include line begins with `$INCLUDE` in column 1, and is followed by a file name. This feature is particularly useful for separating different types of data into multiple files, for example:

```
$INCLUDE /etc/named/data/mailboxes
```

The line is interpreted as a request to load the file `/etc/named/data/mailboxes` at that point. The `$INCLUDE` command does not cause data to be loaded into a different zone or tree. This is simply a way to allow data for a given zone to be organized in separate files. For example, mailbox data might be kept separately from host data using this mechanism.

\$ORIGIN

The origin command is a way of changing the origin in a data file. The line starts in column 1, and is followed by a domain name. It resets the current origin for relative domain names (i.e. not fully qualified names) to the stated name. This is useful for putting more than one domain in a data file.

Resource Record Types

The following are some of the most commonly used types of RR's:

Table 17-1 *Commonly used types of RR's*

Type	Description
SOA	Start of Authority
NS	Name Server
A	Internet Address
CNAME	Canonical Name (nickname)
HINFO	Host Information
WKS	Well Known Services
PTR	Pointer
MX	Mail Exchanger

The following is an example of a hosts file. It is presented here for illustration purposes only. It will be analyzed fully later.

```

;   sample hosts file
@           IN   SOA   ourfox.Sample.Edu. kjd.monet.Sample.Edu. (
                                0290.1506           ; Serial
                                10800           ; Refresh
                                1800           ; Retry
                                3600000        ; Expire
                                86400 )       ; Minimum
                                IN   NS       ourarpa.Sample.Edu.
                                IN   NS       ourfox.Sample.Edu.
ourarpa     IN   A     128.32.0.4
            IN   A     10.0.0.78
            IN   HINFO Sun4/280 UNIX
arpa        IN   CNAME ourarpa
ernie       IN   A     128.32.0.6
            IN   HINFO Sun4/280 UNIX
ourernie    IN   CNAME ernie
monet       IN   A     128.32.7
            IN   A     128.32.130.6
            IN   HINFO Sun-4/280 UNIX
ourmonet    IN   CNAME monet
ourfox      IN   A     10.2.0.78
            IN   A     128.32.0.10
            IN   HINFO Sun-4/280 UNIX
            IN   WKS   128.32.0.10 UDP syslog route timed domain
            IN   WKS   128.32.0.10 TCP ( echo telnet
                                discard sunrpc sftp
                                uucp-path systat daytime
                                netstat qotd nntp
                                link chargen ftp
                                auth time whois mtp
                                pop rje finger smtp
                                supdup hostnames
                                domain
                                nameserver )
fox          IN   CNAME ourfox
toybox      IN   A     128.32.131.119
            IN   HINFO "Sun 4/60"
toybox      IN   MX    0 monet.Sample.Edu
miriam      IN   MB    vineyd.Neighbor.COM.
postmistress IN MR    miriam
bind        IN   MINFO bind-request kjd.Sample.Edu.
            IN   MG    ralph.Sample.Edu.
            IN   MG    zhou.Sample.Edu.
            IN   MG    painter.Sample.Edu.
            IN   MG    riggle.Sample.Edu.
            IN   MG    terry.pa.Xerox.Com.

```

SOA - Start Of Authority

The following is the format of a Start of Authority resource record:

```

name {ttl} {class} SOA      origin  person_in_charge (
                        serial
                        refresh
                        retry
                        expire
                        minimum )

```

The Start of Authority (SOA) record designates the start of a zone. The zone ends at the next SOA record.

name indicates the name of the zone. In the example below, @ indicates the current zone or origin.

IN is the address class

SOA is the type of this Resource Record.

Origin is the name of the host where this data file resides.

person_in_charge is the mailing address for the person responsible for the name server.

Serial is the version number of this data file. You *must* increment this number whenever you make a change to the data: secondary servers use the Serial field to detect whether the data file has been changed since the last time they copied the file from the master server.

No particular format is imposed on you, with the exception that the name server cannot handle numbers over 9999 after the decimal point. However, you should select a format that is rational enough and flexible enough for you to keep track of it. For instance

```
yy.ddd[0-9]
```

where yy stands for the year and ddd stands for the day's number in the year, would permit you up to ten revisions on a given day. You could also use

```
mmyy.dd[0-9][0-9]
```

for up to one hundred revisions per day, with the advantage that you do not have to count days to get the proper number for, say, October 15.

Refresh indicates how often, in seconds, a secondary name server should check with the primary name server to see if an update is needed.

Retry indicates how long, in seconds, a secondary server is to retry after a failure to check for a refresh.

Expire is the upper limit, in seconds, that a secondary name server is to use the data before it expires for lack of getting a refresh.

Minimum is the default number of seconds to be used for the time to live field on resource records that don't have a ttl specified.

There should only be one SOA record per zone.

The following is a sample SOA resource record:

```

;name      {ttl}   class  SOA          origin      person_in_charge
@          3600     IN     SOA          ourfox.Sample.Edu.  kjd.monet.Sample.Edu.(
          0290.1506 ; Serial
          10800    ;Refresh
          3600    ;Retry
          432000  ;Expire
          86400)  ;Minimum

```

NS - Name Server

The following is the format of an NS resource record:

```
{name} {ttl} class NS Name-server name
```

The Name Server record (NS) lists by name a server responsible for a given domain. The name field lists the domain that is serviced by the listed name server. If no name field is listed, then it defaults to the last name listed. One NS record should exist for each primary master server for the domain. The following is a sample NS resource record:

```

;{name} {ttl} class NS Name-server name
          3600     IN     NS     ourarpa.Sample.Edu.

```

A - Address

The following is the format of an A resource record:

```
{name} {ttl} class A address
```

The Address record (A) lists the address for a given machine. The name field is the machine name, and the address is the IP address. One A record should exist for each address of the machine (in other words, gateways should be listed twice, once for each address).

```

;{name} {ttl} class A address
ourarpa      IN    A    128.32.0.4
              IN    A    10.0.0.78

```

HINFO - Host Information

The following is the format of a HINFO resource record:

```
{name} {ttl} class HINFO Hardware OS
```

The Host Information resource record (HINFO) contains host specific data. It lists the hardware and operating system that are running at the listed host. Note that if you want to include a space in the machine name, you must surround the name in double quotes. The name field specifies the name of the host. If no name is specified it defaults to the last named host. One HINFO record should exist for each host. The following is a sample HINFO resource record:

```

;{name} {ttl} class HINFO Hardware OS
              IN    HINFO Sun-3/280 UNIX

```

You can use the *Hardware* and *os* fields to record more information, such as interface and operating system version; the following are some examples:

```

;{name} {ttl} class HINFO Hardware OS
              IN    HINFO Sun-3/280 UNIX-SunOS-4.0.3
              IN    HINFO Sun-4/280 UNIX-SunOS-4.1
              IN    HINFO DEC-uVAXI UNIX-Ultrix-4.0
              IN    HINFO IBM-RT UNIX-AIX-2.2.1
              IN    HINFO IBM-3090/400-knet VM/CMS-XA
              IN    HINFO IBM-3090/400-8232 VM/CMS-XA

```

WKS - Well Known Services

The following is the format of a WKS resource record:

```
{name} {ttl} class WKS address protocol list of services
```

The Well Known Services record (WKS) describes the well known services supported by a particular protocol at a specified address. The list of services and port numbers come from the list of services specified in the *services* database. Only one WKS record should exist per protocol per address. The following is an example of a WKS resource record:

```

;{name} {ttl} class WKS address protocol list of services
IN WKS 128.32.0.10 UDP who route timed domain
IN WKS 128.32.0.10 TCP (echo telnet
discard sunrpc sftp
uucp-path systat daytime
netstat qotd nntp
link chargen ftp
auth time whots mtp
pop rje finger smtp
supdup hostnames
domain
nameserver)

```

CNAME - Canonical Name

The format of a CNAME resource record is as follows:

```
nickname {ttl} class CNAME Canonical name
```

The Canonical Name resource record (CNAME) specifies a nickname for a canonical name (that is the formal or real name). A nickname should be unique. All other resource records should be associated with the canonical name and not with the nickname. Do not create a nickname and then use it in other resource records. Nicknames are particularly useful during a transition period, when a machine's name has changed but you want to permit people using the old name to reach the machine. The following is a sample CNAME resource record:

```

;nickname {ttl} class CNAME Canonical name
ourmonet IN CNAME monet

```

PTR - Domain Name Pointer

The following is the format for a PTR resource record:

```
special name {ttl} class PTR real name
```

A Pointer record (PTR) allows special names to point to some other location in the domain. PTR's are used mainly in the IN-ADDR.ARPA records for the translation of an address (the special name) to a real name. PTR names should be unique to the zone. The PTR records below set up reverse pointers for the special IN-ADDR.ARPA domain.

<i>;special name</i>	<i>{ttl}</i>	<i>class</i>	<i>PTR</i>	<i>real name</i>
7.0		IN	PTR	monet.Podunk.Edu.
2.2.18.128.in-addr.arpa		IN	PTR	blah.junk.COM.

MX - Mail Exchanger

The following is the format for an MX resource record:

```
name {ttl} class mx preference value mailer exchanger
```

The Mail Exchanger (MX) resource records are used to specify a machine that knows how to deliver mail to a domain or machines in a domain. There may be more than one MX resource record for a given name. In the example below, Seismo.CSS.GOV. (note the fully qualified domain name) is a mail gateway that knows how to deliver mail to Munnari.OZ.AU. Other machines on the network cannot deliver mail directly to Munnari. Seismo and Munnari may have a private connection or use a different transport medium. The *preference value* field indicates the order a mailer should follow when there is more than one way to deliver mail to a single machine. The higher the value the lower the preference. The value 0 (zero) indicates the highest preference. Do not use negative numbers. If there is more than one MX resource record for the same name, they may or may not have the same preference value.

You can use names with the wildcard asterisk (*) for mail routing with MX records. There are likely to be servers on the network that simply state that any mail to a domain is to be routed through a relay. In the example below, all mail to hosts in domain foo.COM is routed through RELAY.CS.NET. You do this by creating a wildcard resource record, which states that the mail exchanger for *.foo.COM is RELAY.CS.NET. Note that the asterisk will match any host or sub-domain of foo.COM, but it will not match foo.COM itself.

<i>;name</i>	<i>{ttl}</i>	<i>class</i>	<i>MX</i>	<i>preference value</i>	<i>mailer exchanger</i>
Munnari.OZ.AU.		IN	MX	0	Seismo.CSS.GOV.
foo.COM.		IN	MX	10	RELAY.CS.NET.
*.foo.COM.		IN	MX	20	RELAY.CS.NET.

MB - Mailbox

The following is the format for an MB resource record:

```
name {ttl} class MB Machine
```

Note: The *MB*, *MR*, *MINFO*, and *MG* records are not implemented at the present time. They are described for informational purposes only.

The Mailbox record (MB) lists the machine where a user wants to receive mail. The name field contains the user's login name; the machine field denotes the machine where mail should be delivered. Mailbox names should be unique to the zone. The following is a sample MB resource record:

```
;name      {ttl}  class  MB  Machine
miriam          IN    MB  vineyd.Neighbor.COM.
```

MR - Mail Rename Record

The following is the format for the MR resource record:

```
name      {ttl}  class  MR  corresponding MB
```

You use the Mail Rename (MR) record to list aliases for a user. The name field lists the alias for the name listed in the fourth field, which should have a corresponding MB record. The following is a sample MR resource record that specifies that mail received for "postmistress" should be routed to "miriam" at the machine specified in the MB resource record for that name.

```
;name      {ttl}  class  MR  corresponding MB
postmistress      IN    MR  miriam
```

MINFO - Mailbox Information

The following is the format for the MINFO resource record:

```
name      {ttl}  class  MINFO  requests  maintainer
```

The Mail Information record (MINFO) creates a mail group for a mailing list. This resource record is usually associated with at least one Mail Group resource record, but may be used with a Mail Box record. The *name* field specifies the name of the mailbox. The *requests* field indicates where to send mail such as requests to be added to a mail group. The *maintainer* field specifies the mailbox to receive error messages. This field is particularly important for mailing lists that require errors in members names to be reported to a person other than the sender. The following is a sample MINFO resource record:

```
;name      {ttl}  class  MINFO  requests  maintainer
bind          IN    MINFO  bind-request  kjd.Sample.Edu.
```

MG - Mail Group Member

The following is the format for an MG resource record:

```
{name} {ttl} class MG member name
```

The Mail Group record (MG) lists members of a mail group. The following is a sample MG resource record:

```
;{name} {ttl} class MG member name
                IN      MG      ralph.Sample.Edu.
```

An example for setting up a mailing list is as follows:

```
bind IN MINFO bind-request kjd.Sample.Edu.
      IN MG      ralph.Sample.Edu.
      IN MG      zhou.Sample.Edu.
      IN MG      painter.Sample.Edu.
      IN MG      riggle.Sample.Edu.
      IN MG      terry.pa.Xerox.Com.
```

17.6. A Practical Example

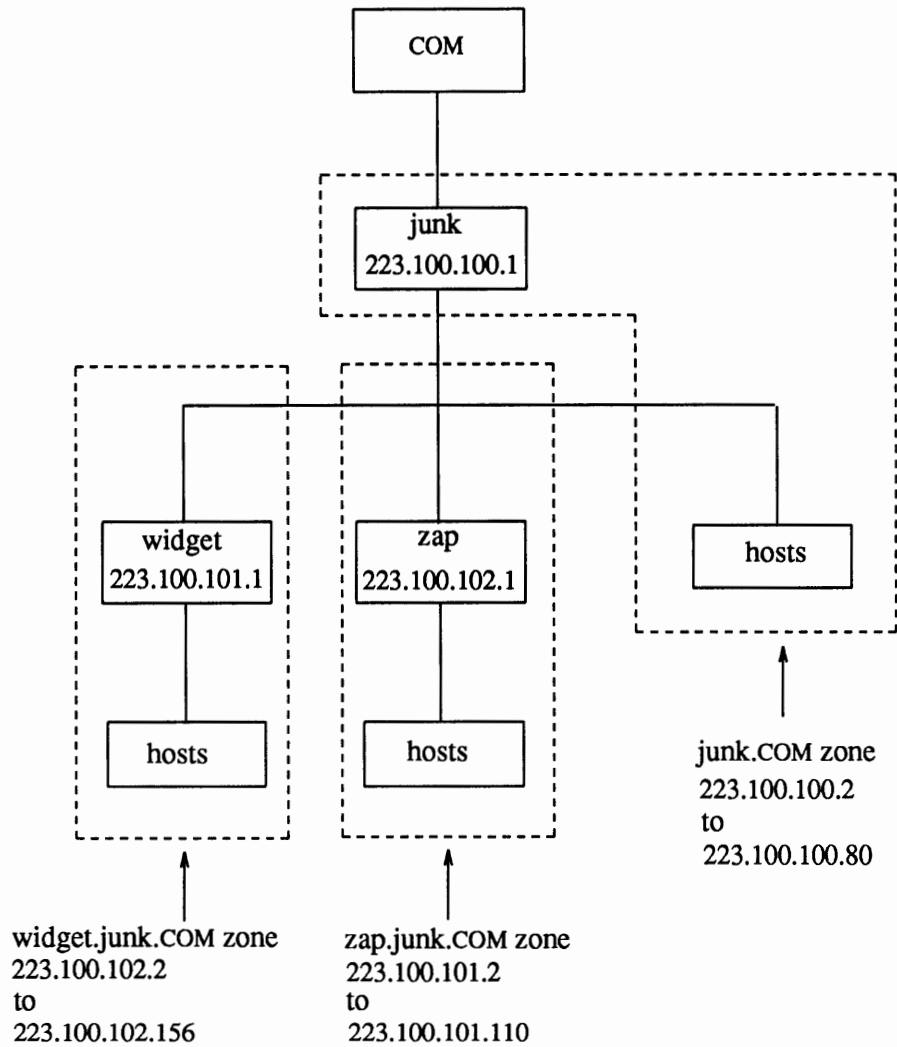
Given the above information, you can now start building up the files that an *imaginary* network would need. Assume that your network is composed of three networks, each with a Class C Network Number:

```
name    number
junk    223.100.100
widget  223.100.101
zap     223.100.102
```

The names of the zones are also the names of the hosts being designated as the master servers.

This imaginary network can therefore be represented by the following figure:

Figure 17-4 An imaginary network



Assume further that after careful consideration you decide to set up the Domain Name Service in network so that each master server is the primary server for its zone and a secondary server for the other zones. All this can be summed up by the following tables:

junk zone		
hostname	function	address
junk	primary	223.100.100.1
widget	secondary	223.100.101.1
zap	secondary	223.100.102.1
hosts	hosts	223.100.100.2-80

widget zone		
<i>hostname</i>	<i>function</i>	<i>address</i>
widget	primary	223.100.101.1
junk	secondary	223.100.100.1
zap	secondary	223.100.102.1
	hosts	223.100.101.2-110

zap zone		
<i>hostname</i>	<i>function</i>	<i>address</i>
zap	primary	223.100.102.1
junk	secondary	223.100.100.1
widget	secondary	223.100.101.1
	hosts	223.100.102.2-156

The following are the boot files for the three servers in the network:

```

;
;  Boot file for server junk

directory  /var/named
cache      .                named.root
primary    junk.COM         junk.zone
primary    100.100.223.in-addr.arpa  junk.revzone
primary    0.0.127.in-addr.arpa      named.local

secondary  widget.junk.COM      223.100.101.1 223.100.102.1 widget.zone
secondary  zap.junk.COM        223.100.101.1 223.100.102.1 zap.zone
secondary  101.100.223.in-addr.arpa 223.100.101.1 widget.rev
secondary  102.100.223.in-addr.arpa 223.100.102.1 zap.rev

```

```

;
;  Boot file for server widget

directory  /var/named
cache      .                named.root
primary    widget.junk.COM   widget.zone
primary    101.100.223.in-addr.arpa  widget.revzone
primary    0.0.127.in-addr.arpa      named.local

secondary  junk.COM          223.100.100.1 223.100.102.1 junk.zone
secondary  zap.junk.COM      223.100.100.1 223.100.102.1 zap.zone
secondary  100.100.223.in-addr.arpa 223.100.100.1 junk.rev
secondary  102.100.223.in-addr.arpa 223.100.102.1 zap.rev

```

```

;
;  Boot file for server zap

directory  /var/named
cache      .                named.root
primary    zap.junk.COM     zap.zone
primary    101.100.223.in-addr.arpa  zap.revzone
primary    0.0.127.in-addr.arpa      named.local

secondary  junk.COM          223.100.100.1 223.100.101.1 junk.zone
secondary  widget.junk.COM   223.100.100.1 223.100.101.1 widget.zone
secondary  100.100.223.in-addr.arpa 223.100.100.1 junk.rev
secondary  101.100.223.in-addr.arpa 223.100.101.1 widget.rev

```

The following are some sample `resolv.conf` files. Note that if the host in question is not running `named` the local host address should not be used as a `nameserver`.

```
;
; resolv.conf file for server junk
;
domain      junk.COM
nameserver  127.0.0.1
nameserver  223.100.101.1
nameserver  223.100.102.1
```

```
;
; resolv.conf file for a host in zone junk not running named
;
domain      junk.COM
nameserver  223.100.100.1
nameserver  223.100.101.1
nameserver  223.100.102.1
```

```
;
; resolv.conf file for a host in zone widget.junk not running named
;
domain      widget.junk.COM
nameserver  223.100.100.1
nameserver  223.100.101.1
nameserver  223.100.102.1
```

```
;
; resolv.conf file for a host in zone zap.junk not running named
;
domain      zap.junk.COM
nameserver  223.100.100.1
nameserver  223.100.101.1
nameserver  223.100.102.1
```

The following are sample local host files for the servers:

```

;
; named.local for server junk
;
@      IN      SOA      junk.COM.      ralph.sysad.zap.junk.COM. (
        0290.1506 ;Serial
        10800     ;Refresh
        3600      ;Retry
        432000    ;Expire
        86400)    ;Minimum

1      IN      NS       junk.COM.
1      IN      PTR      localhost.

```

```

;
; named.local for server widget
;
@      IN      SOA      widget.junk.COM. ralph.sysad.zap.junk.COM. (
        0290.1506 ;Serial
        10800     ;Refresh
        3600      ;Retry
        432000    ;Expire
        86400)    ;Minimum

1      IN      NS       widget.junk.COM.
1      IN      PTR      localhost.

```

```

;
; named.local for server zap
;
@      IN      SOA      zap.junk.COM.    ralph.sysad.zap.junk.COM. (
        0290.1506 ;Serial
        10800     ;Refresh
        3600      ;Retry
        432000    ;Expire
        86400)    ;Minimum

1      IN      NS       zap.junk.COM.
1      IN      PTR      localhost.

```

The following is the host file for server junk, followed by its \$INCLUDE'd file:

```
;
; junk zone hosts file for server junk
;
@           IN      SOA      junk.COM.          ralph.sysad.zap.junk.COM. (
                0290.1506 ;Serial
                10800    ;Refresh
                3600     ;Retry
                432000   ;Expire
                86400)   ;Minimum

                IN      NS      junk.COM.
                IN      NS      widget.junk.COM.
widget.junk.COM. IN      NS      zap.junk.COM.
                IN      NS      widget.junk.COM.
                IN      NS      junk.COM.
zap.junk.COM.   IN      NS      zap.junk.COM.
                IN      NS      zap.junk.COM.
                IN      NS      junk.COM.
                IN      NS      widget.junk.COM.
junk.COM.      IN      MX      10 junk.COM.
*.junk.COM.    IN      MX      10 junk.COM.

; junk.COM hosts
$INCLUDE /var/named/hosts/junk
```

```

; hosts in junk zone as listed in /var/named/hosts/junk
junk          A      223.100.100.1
              A      10.1.0.56
              MX     10 junk.COM.
widget        A      223.100.101.1
              HINFO  "Sun 3/180"   Unix
              MX     10 junk.COM.
              WKS    223.100.101.1  UDP syslog timed domain
              WKS    223.100.101.1  TCP (echo telnet
              discard sunrpc sftp
              uucp-path systat daytime netstat
              qotd nntp link chargen ftp auth
              time whots mtp pop rje finger
              smtp supdup hostnames
              domain nameserver)
zap           A      223.100.102.1
              HINFO  Sun-4/280   Unix
              MX     10 junk.COM.
              WKS    223.100.102.1  UDP syslog timed domain
              WKS    223.100.102.1  TCP (echo telnet
              discard sftp
              uucp-path systat daytime netstat
              qotd nntp link chargen ftp auth
              time whots mtp pop rje finger
              smtp supdup hostnames
              domain nameserver)
lazy         A      223.100.100.2
              HINFO  "Sun 3/50"   Unix
              MX     10 junk.COM.
crazy        A      223.100.100.3
              HINFO  Sun3/60     Unix
              MX     10 junk.COM.
hazy         A      223.100.100.4
              HINFO  Sun3/80     Unix
              MX     10 junk.COM.

; all other hosts follow, up to 223.100.100.80

```

The following is the host file for server widget, followed by its \$INCLUDE'd file:

```

;
; widget zone hosts file for server widget
;
@      IN      SOA      widget.junk.com.  ralph.sysad.zap.junk.com. (
        0290.1506  ;Serial
        10800     ;Refresh
        3600      ;Retry
        432000    ;Expire
        86400)    ;Minimum

        IN      NS      widget.junk.com.
        IN      NS      junk.com.
        IN      NS      zap.junk.com.

; junk.COM hosts
$INCLUDE /var/named/hosts/widget

```

```

; hosts in widget zone as listed in /var/named/hosts/widget

widget      A      223.100.101.1
            HINFO  "Sun 3/180"      Unix
            MX     10 junk.COM.

whatsit     CNAME  widget.junk.COM
smelly      A      223.100.101.2
            HINFO  Sun3/50        Unix
            MX     10 junk.COM.

stinky      A      223.100.101.3
            HINFO  Sun4/280      Unix
            MX     10 junk.COM.

dinky       A      223.100.101.4
            HINFO  "Sun 3/160"    Unix
            WKS    223.100.101.4  UDP who route
            WKS    223.100.101.4  TCP (echo telnet
                                   discard sftp
                                   uucp-path systat daytime netstat
                                   ftp finger domain nameserver)

            MX     10 junk.COM.

; all other hosts follow, up to 223.100.101.110

```


The following is the sample file of reverse addresses for hosts in the zone junk. Note that the name of the domain is fully qualified, so that the addresses of the hosts (without the network address) is sufficient in this case:

```

; reverse address file for server junk, in /var/named/junk.revzone
100.100.223.in-addr.arpa.  IN  SOA          junk.COM.          ralph.sysad.zap.junk.COM. (
                                0290.1506         ;Serial
                                10800          ;Refresh
                                3600           ;Retry
                                432000        ;Expire
                                86400)        ;Minimum

                                IN  NS            junk.COM.
1                                PTR            junk.COM.
2                                PTR            lazy.junk.COM.
3                                PTR            crazy.junk.COM.
4                                PTR            hazy.junk.COM.

; all other hosts follow, up to 223.100.100.80

```

The reverse address files for servers widget and zap should be written in a manner similar to the above.

Adding a Cache Only Server

You could also add a cache only server to our imaginary set up. Zone bond.junk.com could be served by host bond, at 223.100.103.1, and have hosts in the range 223.100.103.2 to 223.100.103.80. Its named.boot file would appear as shown below:

```

;
;  Boot file for server bond
directory /var/named
cache     .          named.root

```

All other files would be similar to the ones corresponding to the other servers and hosts in the example.

Self-contained DNS

If your network is not connected to the Internet, that is, it's a self-contained domain with no connection to the outside world, you can also run DNS on it, but you must remember that DNS was conceived and implemented with the Internet in mind, and running it in any other way can never be anything but a hack. The following example is presented only for illustration purposes, and not as a proper or recommended way of running DNS.

DNS expects a well formed name space at all times, that is, domain names must conform to the rules outlined in this chapter, even if your domain is not connected to the top level. A fully qualified domain name must, therefore, start at the root (.) domain.

The strategy is to give "root" responsibility to the primary server for the domain, in this case junk, and to create a root cache file that points back to the same server.

Modify the named.boot file for the primary server:

```

;
;  Boot file for server junk

directory  /var/named
cache      .                named.root
primary    .                root.zone
primary    junk.COM        junk.zone
primary    100.100.223.in-addr.arpa  junk.revzone
primary    0.0.127.in-addr.arpa  named.local

secondary  widget.junk.COM    223.100.101.1 223.100.102.1 widget.zone
secondary  zap.junk.COM      223.100.101.1 223.100.102.1 zap.zone
secondary  101.100.223.in-addr.arpa  223.100.101.1 widget.rev
secondary  102.100.223.in-addr.arpa  223.100.102.1 zap.rev

```

Now create a new named.root file:

```

;
;  named.root - initial cache data for root domain servers
;
;
;  list of servers...
.          99999999          IN      NS      junk.COM.
;
;
;  ...and their addresses
junk.COM.  99999999          IN      A      223.100.100.1

```

Now create a file for the root server:

```

;
; root.zone - host file for root server
;
@           IN      SOA      junk.COM.      root.junk.COM. (
                0290.1506 ;Serial
                10800    ;Refresh
                3600     ;Retry
                432000   ;Expire
                86400)   ;Minimum

                IN      NS      junk.COM.
junk.COM      IN      A        223.100.100.1
1.100.100.223.in-addr.arpa. IN PTR      junk.COM.

```

All other files should essentially remain unchanged. You have simply created a loop back to your own server, which is also set up as a root server. This, or variations thereof, should work quite well as long as you remember that now DNS will only be able to serve queries within its single domain authority.

17.7. Setting Up DNS

This section contains information for starting `named` and the resolver.

Starting `named`

The presence of a file called `/etc/named.boot` automatically triggers the invocation of the `in.named` daemon. This is controlled in the `/etc/rc.local` file by the lines:

```

if [ -f /usr/etc/in.named -a -f /etc/named.boot ]; then
    in.named;          echo -n ' named'
fi

```

CAUTION: Do not attempt to run `named` from `inetd`. This will continuously restart the name server and defeat the purpose of having a cache.

The above lines assume that the boot file you are using in this server is called `/etc/named.boot`. If you are using a different name, modify the above lines to reflect the new name, and also provide that name in the line that invokes `named` (see the man page for `in.named`). For instance, if the name of the file is `/etc/named.start` then you would have to modify the above lines to:

```

if [ -f /usr/etc/in.named -a -f /etc/named.start ]; then
    in.named -b /etc/named.start;      echo -n ' named'
fi

```

Starting the resolver

In each machine that is to run the resolver, create the appropriate `/etc/resolv.conf` file.

The resolver is comprised of a few routines that build query packets and exchange them with the name server. Normally only the NIS server needs to be linked directly with the resolver library, and other programs use the normal NIS functions to access names. The above is accomplished by using the `-b` flag in

the Makefile located in `/var/yp` on the NIS master server (see *The Network Information Service*). However, a version of `/usr/lib/sendmail` directly linked with the resolver is also provided, (as `/usr/lib/sendmail.mx`) since mail delivery involves more information than simple name to address mapping, such as access to Mail Exchanger (MX) information. Using `sendmail.mx` might also involve modifying the `/etc/sendmail.cf` file on the mail host.

17.8. Modifying the database

When you add or delete a host in one of the data files in the master DNS server, or modify otherwise the data files, you must also change the Serial number in the SOA resource record so the secondary servers modify their data accordingly; you should then inform `named` in the master server that it should re-read the data files and update its internal database, as explained below.

named's PID

When `named` successfully starts up, it writes its process ID to the file `/etc/named.pid`. Thus you do not have to run `ps` to obtain `named`'s process ID; just running `cat` on `named.pid` will be enough (and it will be faster too).

Reload (SIGHUP)

To have `named` re-read `named.boot` and reload the database that you have just changed, enter:

```
# kill -HUP `cat /etc/named.pid`
```

Note that all previously cached data is lost, and the caching process starts anew.

17.9. Debugging `named`

You can also debug `named` by sending it signals through the `kill(1)` utility. Depending on the signal received `named` will change its behavior.

Database Browsing (SIGINT)

To get a good idea of what `named` thinks the database is, enter:

```
# kill -INT `cat /etc/named.pid`
```

Upon receiving this signal, `named` dumps the current data base and cache to `/var/tmp/named_dump.db`. This should give you an indication of whether the database was loaded correctly. If you have source code, you can change the name of the dump file by re-defining `DUMPFIL` to the new name when compiling `named`.

When `named` is running incorrectly, you can also look in `/usr/adm/messages` and check for any messages logged by `syslog`. For instance, if there is a data file that lists a hostname as a nickname and also has other data under that name instead of the machine's canonical name, you may see a message similar to:

```
May 4 02:35:26 hostname named[4804]: hazy.widget.junk.com
has CNAME and other data (illegal)
```

Or, if there is a problem with the database, you may see:

```
May 1 11:02:33 hostname named[17808]: /etc/named/junk.zone:
line 759: database format error ()
```

Turning on debugging (SIGUSR1)

To turn on debugging, you can start `named` with the `-d` option, or, if `named` is already running, you can enter:

```
# kill -USR1 `cat /etc/named.pid`
```

Each following `USR1` increments the debug level. The output goes to `/var/tmp/named.run`.

Turning off debugging (SIGUSR2)

To turn off debugging completely, enter:

```
# kill -USR2 `cat /etc/named.pid`
```

Using `nslookup`

The `nslookup` utility is an interactive program that you can use to query Internet domain name servers. You can contact servers to request information about a specific host or print a list of hosts in the domain.

When you initially enter the command `nslookup` you will see a message similar to:

```
Default server: localhost.junk.com
Address: 127.0.0.1

>
```

The `>` is `nslookup`'s prompt. Whether your default server is `localhost` or some other server will depend on how you have set the DNS and on the contents of the `/etc/resolv.conf` file in the server you are using.

If you want to find the address, for instance, of host `lazy` in the domain `junk.com` you can simply enter its name at the prompt:

```
> lazy
```

and the address will be found if there is indeed such a host. You can also enter the fully qualified name. If the host you are looking for is not in this domain, then you must enter the fully qualified name.

You can also see the kind of queries sent by the server, and the answers it gets to these queries when you ask for information, by entering:

```
> set debug
```

The manual page for `nslookup(8c)` has a complete description of all the other options.

17.10. Administering DNS for Your Domain

Setting up DNS involves not only running the appropriate programs on the servers and clients, but also determining domain names, answering complaints, and filling out registration and other forms, should you join a public network. This section contains overall procedures for setting up DNS, based on the responsibilities you might have as a DNS administrator.

Types of Administrators and Their Responsibilities

Your administrative responsibilities for DNS depend on your domain's position within the overall network hierarchy. For example, managing one set of name servers in a small administrative domain entails less responsibility than managing the authoritative set for a large zone. Responsibilities depend on whether you are the chief authority for a domain or zone, or an administrator reporting to the chief authority.

The NIC divides administrators on the Internet into domain administrators—chief authorities—or technical contacts—administrators reporting to the chief. Descriptions of each position appear below. Read them regardless of the public network you belong to (if any), determine which best describes your function, then follow the appropriate procedures for your position.

The Domain Administrator

The domain administrator (DA) is a coordinator, manager, and technician for a second level or lower domain. The DA's responsibilities therefore include:

- Registering the domain. If your domain is going to be on a public network, you must register it if you have not done so already. The domain must have a unique name within its level of the network hierarchy. Contact the organization in charge of the network, and request the appropriate domain registration form.

To belong to multiple networks such as CSNET, the Internet, and BITNET, register with only one network, since domain names are independent of a network's physical topology.

Appendix F, *IP Number Registration Form*, contains appropriate information for registering your domain with the Internet.

If you are a CSNET member organization and have not registered your domain, contact the CSNET Coordination and Information Center (CIC). Request an application and information about setting up a domain.

If your organization has already registered its domain name with the CIC, you should keep them informed about how you want your mail routed. In general, the CSNET relay prefers to send mail via CSNET (as opposed to over the Internet or via uucp). For an organization on multiple networks, this may not always be the preferred behavior. You can reach the CIC via electronic mail at `cic@sh.cs.net`, or by phone at (617) 497-2777.

If you are on the BITNET and need to set up a domain, contact `INFO@BITNIC` or send mail to `domains%bitnic.bitnet@cunyvm.cuny.EDU`; you can also obtain information by sending the message `SEND DOMAIN GUIDE TO listserv%bitnic.bitnet@cunyvm.cuny.EDU`.

- Naming hosts and verifying that names within the domain are unique.

At many sites, users name their individual hosts while the administrators name the servers. The administrator should ensure that there are no duplicate names within a zone. This can be done either by inspecting all the resource records for the zone, or by using the `nslookup` program. This is an interactive program that you can use (among other things) either to produce a listing of all the hosts in the zone or domain (by using the `ls` command at the `>` prompt), or to obtain information about a host: at the prompt you enter the host's name; if the host is in the database, `nslookup` will produce its address, otherwise it will inform you that it does not exist. (Please refer to the man page for `nslookup(8)` for a detailed explanation of this command.)

If you find duplicate host names, ask the hosts' users to provide different names. Later, when you add new hosts, look up a prospective host name with `nslookup` to verify that it is unique.

- Understanding the functioning of the name servers and making sure the data is current at all times. This includes either setting up the DNS-related files and programs, or delegating this authority to other technically competent people (the technical/zone contact).

This chapter should provide you with the basic information you need for understanding DNS. However, as the Domain Administrator, you should gain more specific technical knowledge of your particular network.

- Read the man pages for `named(8)`, `resolver(3)`, `netconfig(5)` and `resolv.conf(5)` to understand these programs and files.
- Contact the public network to which you belong and ask for technical papers regarding it.
- *Getting Internet Information from the NIC*

The NIC offers a large selection of technical papers about the Internet and how it relates to TCP/IP, electronic mail, and other services. These papers are called *RFCs* (Request For Comment) and are in the public domain. You can receive free copies of the RFCs you need by accessing the NIC's server `NIC.DDN.MIL` and using the `ftp` command to copy them.

- Handling complaints and questions from users.
- Being aware of the behavior of hosts on the domain, in case of security problems, protocol violations, and other misuse of the network.
- Interacting with DAs at the next higher level to solve network-related problems.

The Technical Contact

The domain technical/zone contact is responsible for keeping the name servers running and maintaining the DNS programs and files. Technical contacts also interact with their Domain Administrators and DAs of other domains to solve

network problems.

The main responsibility for a technical contact is the maintenance of the files relating to the corresponding zone.

17.11. Acknowledgements

This chapter is based on RFCs 1032, 1033, 1034 and 1035, and on the "Name Server Operations Guide for BIND, Release 4.8" from the University of California at Berkeley. The following acknowledgment notice appears in the latter.

Many thanks to the users at U.C. Berkeley for falling into many of the holes involved with integrating BIND into the system so that others would be spared the trauma. I would also like to extend gratitude to Jim McGinness and Digital Equipment Corporation for permitting me to spend most of my time on this project.

Ralph Campbell, Doug Kingston, Craig Partridge, Smoot Carl-Mitchell, Mike Muuss and everyone else on the DARPA Internet who has contributed to the development of BIND. To the members of the original BIND project, Douglas Terry, Mark Painter, David Riggle and Songnian Zhou.

Anne Hughes, Jim Bloom and Kirk McKusick and the many others who have reviewed this paper giving considerable advice.

This work was sponsored by the Defense Advanced Research Projects Agency (DoD), Arpa Order No. 4871 monitored by the Naval Electronics Systems Command under contract No. N00039-84-C-0089. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing official policies, either expressed or implied, of the Defense Research Projects Agency, of the US Government, or of Digital Equipment Corporation.

17.12. References

[Birrell]

Birrell, A. D., Levin, R., Needham, R. M., and Schroeder, M.D., "Grapevine: An Exercise in Distributed Computing." In *Comm. A.C.M.* 25, 4:260-274 April 1982.

[RFC 819]"

Su, Z. Postel, J., "The Domain Naming Convention for Internet User Applications." *Internet Request For Comment 819* Network Information Center, SRI International, Menlo Park, California. August 1982.

[RFC 973]

Mockapetris, P., "Domain System Changes and Observations." *Internet Request For Comment 973* Network Information Center, SRI International, Menlo Park, California. February 1986.

[RFC 974]

Partridge, C., "Mail Routing and The Domain System." *Internet Request For Comment 974* Network Information Center, SRI International, Menlo Park, California. February 1986.

[RFC 1032]

Stahl, M., "Domain Administrators Guide" *Internet Request For Comment 1032* Network Information Center, SRI International, Menlo Park, California. November 1987.

[RFC 1033]

Lottor, M., "Domain Administrators Guide" *Internet Request For Comment 1033* Network Information Center, SRI International, Menlo Park, California. November 1987.

[RFC 1034]

Mockapetris, P., "Domain Names - Concept and Facilities." *Internet Request For Comment 1034* Network Information Center, SRI International, Menlo Park, California. November 1987.

[RFC 1035]

Mockapetris, P., "Domain Names - Implementation and Specification." *Internet Request For Comment 1035* Network Information Center, SRI International, Menlo Park, California. November 1987.

[Terry]

Terry, D. B., Painter, M., Riggle, D. W., and Zhou, S., *The Berkeley Internet Name Domain Server*. Proceedings USENIX Summer Conference, Salt Lake City, Utah. June 1984, pages 23-31.

[Zhou]

Zhou, S., *The Berkeley Internet Name Domain Server*. Proceedings USENIX Summer Conference, Salt Lake City, Utah. June 1984, pages 23-31.

[Zhou]

Zhou, S., *The Design and Implementation of the Berkeley Internet Name Domain (BIND) Servers*. UCB/CSD 84/177. University of California, Berkeley, Computer Science Division. May 1984.

vns@
sabadasv@
sabadaszka@ecc2.
uq9980@engtest.

TO: vns@elms-esd.northgrun.com
cc: vns@dsd.

sabadasv

sabadaszka@ecc2.e

uq9980@engtest.esd.

The Remote File Sharing Service

The Remote File Sharing service (RFS) provides resource sharing for hosts on a network, enabling users to access remote resources as though the resources were available locally. RFS for SunOS Release 4.1 is based on the RFS feature of UNIX System V, Release 3 operating system. You can select RFS as an optional software package from SunInstall.

This chapter explains how to set up and administer RFS services on your SunOS network, including these topics.

- Introducing RFS operation in the SunOS network environment
- Installing RFS
- Setting up the RFS name server
- Setting up the RFS file server
- Advertising resources
- Mounting resources
- Setting up RFS Security
- Maintaining RFS services

18.1. RFS in the SunOS Network Environment-an Overview

RFS is a network service that runs on TCP/IP-based networks, among others. You optionally provide RFS services for your SunOS network, along with the default NFS services. However, RFS provides a somewhat different functionality than NFS, as described below.

- Provides sharing of entities called *resources*. By specifying a directory hierarchy as a resource, you enable users to access these directories without having to know their location.
- Enables users to share both regular files and remote devices.
- Provides additional security through user and group ID mapping.
- Can be started and stopped for maintenance or in case a server goes down.
- Allows users to access all file hierarchies of a file system, including those hierarchies mounted below any mount points. When you mount a file system through NFS, you can access all subdirectories except those mounted

below mount points. This type of subdirectory is not accessible to NFS clients unless you separately export it.

- Cannot provide diskless clients with the file systems they need to boot up.
- Does not support symbolic links or automounting.
- Does not allow you to mount a directory hierarchy on top of an already-mounted RFS directory hierarchy.

RFS has its own special terminology and concepts. Moreover, in a SunOS environment, machines that run RFS are classified according to the following types.

- RFS file servers
- RFS clients
- Primary RFS name servers
- Secondary RFS name servers

A group of these machines administered as a unit is considered an RFS domain. The remainder of this overview describes these machine types and the functionality they provide.

RFS File Servers and Resource Sharing

RFS enables machines on a network to mount resources offered by RFS file servers. These resources include not only directory hierarchies, but also remote device files, such as a tape or disk drive, and named pipes. This text refers to any machine that shares a resource via RFS as an *RFS file server*, even if it also shares remote devices or named pipes.

The RFS file server announces the availability of its resources by *advertising* them to the network. To set up advertising, you log in to the file server, give resources their *resource names*, then advertise them by issuing the `adv` command. You can use resource names to disguise the pathname of a resource, and, more importantly, the machine it is on. Additionally, you can assign access rights to the RFS file server to trusted users and groups.

If you wish, you can set up home directories on an RFS file server and have clients mount them as resources, rather than as directory hierarchies. This makes the location of home directories invisible to users.

Diskless and dataless clients cannot rely on RFS services to mount the remote file systems that they need during booting (`/export/exec` and `/usr`). This is because RFS cannot run at this stage of the booting process. However, using SunInstall, you can configure a file server to export these directories to clients through NFS, while providing other resources through RFS.

RFS Clients and Mounting

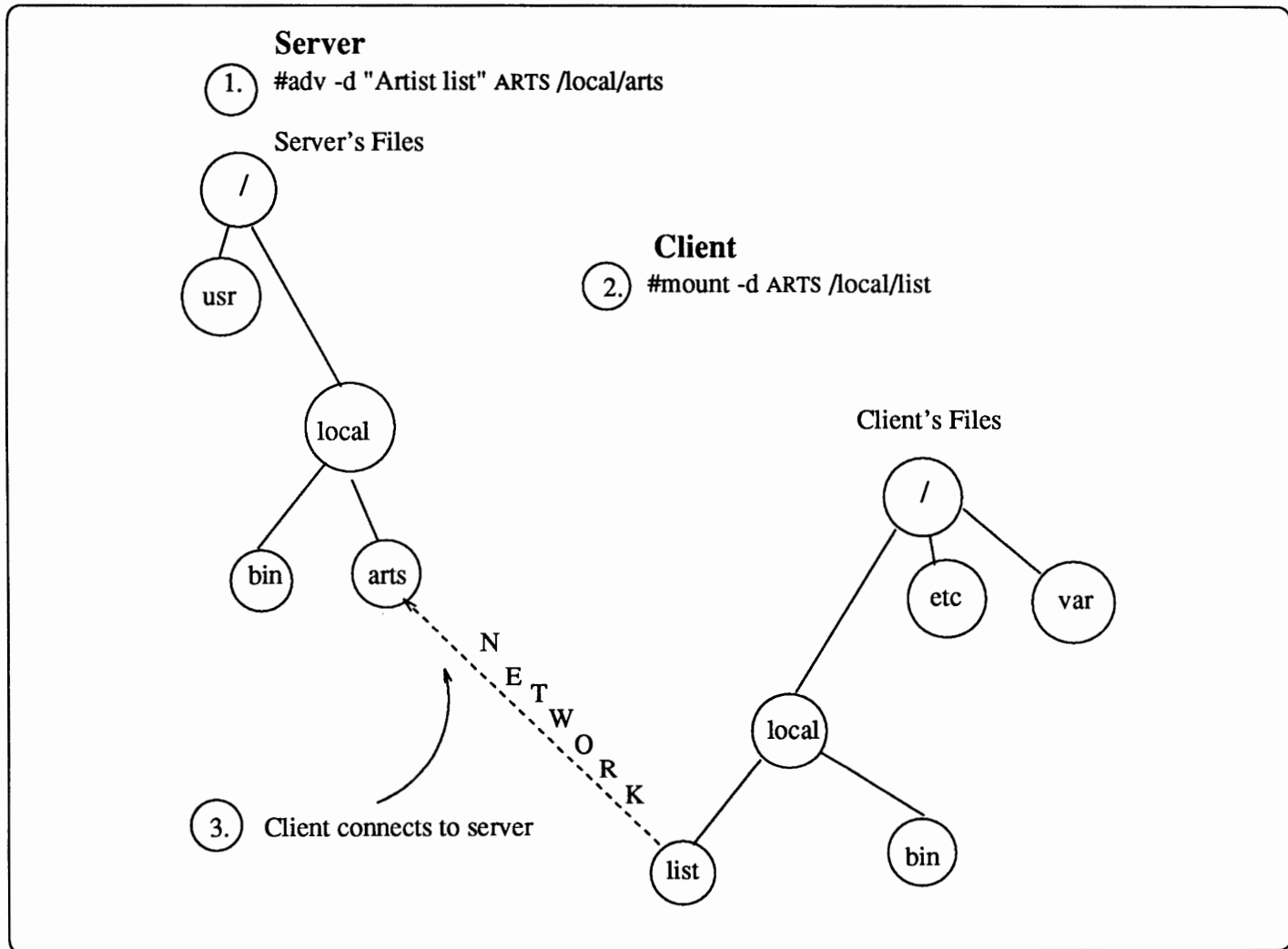
When you use RFS, clients on your network that run Release 4.1 can mount files from NFS or RFS servers, or both. Machines using RFS and NFS can exist in any combination; RFS file servers can be NFS clients, or vice versa.

The `/bin/nsquery` command displays a list of resources available from RFS servers on the network. With this information, a user can now access resources from the RFS server by running the `mount` command with RFS-specific options.

How Resource Sharing Takes Place

Figure 18-1 shows how an RFS client mounts a resource from an RFS file server.

Figure 18-1 Resource Sharing Between an RFS Client and File Server



Here is a description of the events shown in the figure. The administrator of the RFS file server jazz decides to make available the directory hierarchy /local/arts. To do so, the administrator types the following on the file server.

```
# adv -d "Artist list" ARTS /local/arts
```

The following is a breakdown of this command and its arguments.

adv -d

Sets up advertising by announcing the directory's availability to the network, and by sending the information in its arguments to the RFS name server.

"Artist list"	Defines the contents of the resource. This is the description the user will receive from the <code>nsquery</code> command.
ARTS	This is the resource name you give the directory hierarchy.
/local/arts	This is the pathname of the directory hierarchy to be shared under the resource name ARTS .

Assume a user on client `raks` wants to mount an address list of artists, but does not know where to find this list. The user issues `nsquery` with the following results:

```
# nsquery
RESOURCE      ACCESS      SERVER      DESCRIPTION
ARTS          /read/write jazz       Artist list
```

The user sees the resource's name, ARTS, its access rights, server of origin, and a description. However, the user does not see the actual pathname of resource ARTS. As a result, the user on `raks` types the following to mount the resource.

```
# mount -d ARTS /local/list
```

This command tells `mount` that the client wants to access resource ARTS through its mount point `/local/list`. Provided it has the appropriate permission, client `raks` can then perform the mount operation. With RFS services, `mount` uses the underlying protocol (TCP, by default) to set up a virtual circuit connection from the client mount point `/local/list` to `/local/arts` on server `jazz`. The user on client `raks` accesses these files as he or she would access local or NFS-mounted file systems.

Another RFS feature that comes into play during file sharing is *user and group ID mapping*. You set up this optional function on the RFS file server by defining the permissions particular users and groups will have when they mount the file server's resources. A later section on RFS security explains this form of mapping in detail.

The RFS Name Server and its Functions

The *RFS name server* is the machine where you initially start RFS for an RFS domain, and maintain information about that domain. It maintains information about all resources available from RFS file servers. The RFS name server receives this information when the administrator of an RFS file server uses `adv`. The name server also maintains lists of all RFS passwords used in its domain.

The RFS name server provides a service called *name-to-resource mapping*. When a user issues a mount request on a client, it appears that the request goes to the RFS file server. In reality, the RFS name server transparently handles the request. It performs name-to-resource mapping, using the resource name in the `mount` command as the key to find the file server and real pathname of the resource.

The RFS Domain

The term *RFS domain* refers to a group of machines using RFS services that are centralized and administered on the same RFS name server, such as these.

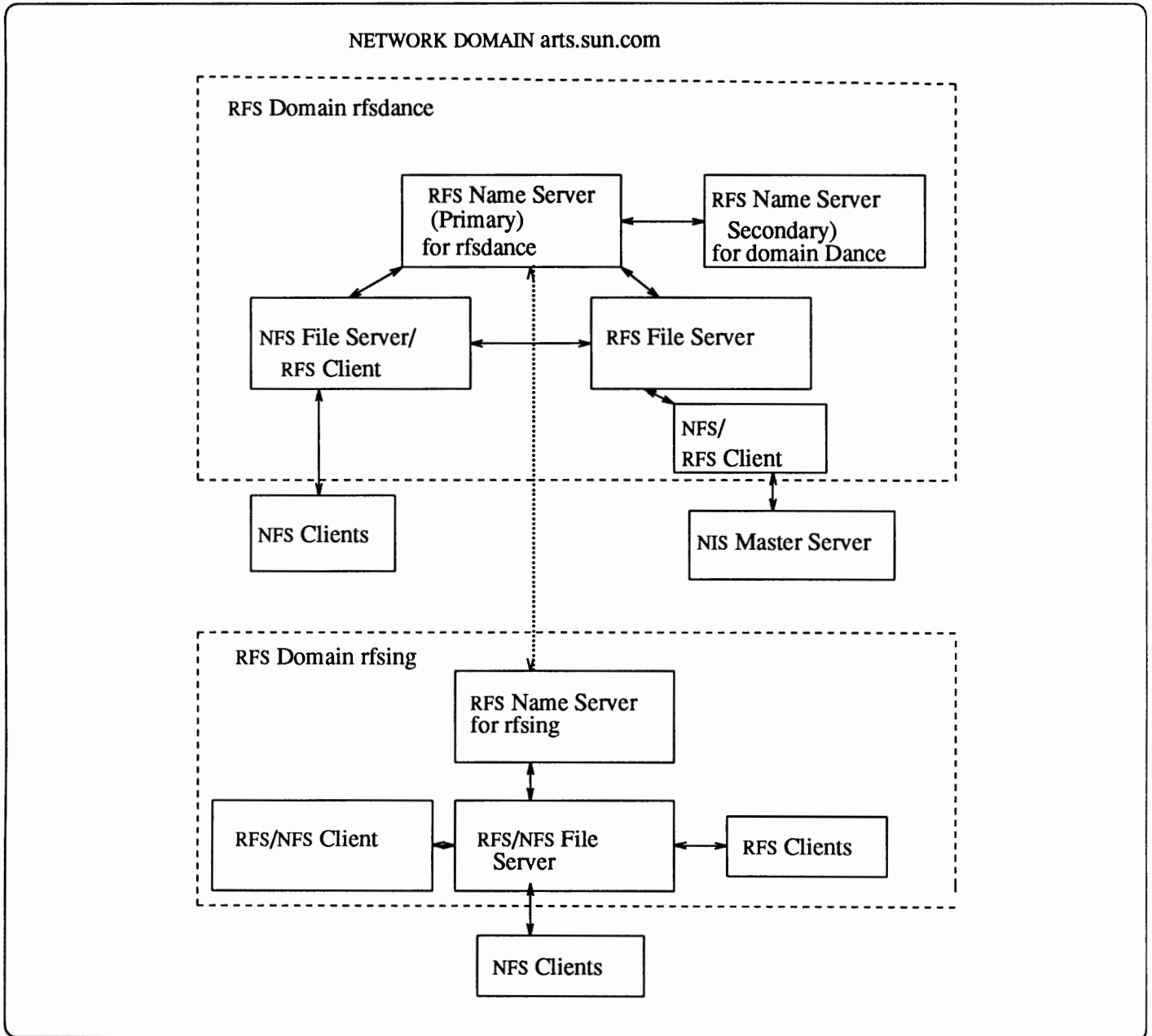
- A primary name server and its tables. The primary name server acts as the central point of access for the RFS domain. Its tables contain all information needed to locate resources, set up security, and communicate with name servers of other RFS domains.
- Possibly one or more secondary name servers, which maintain the same set of tables as the primary. Should the primary fail, a secondary server will immediately take over.
- The RFS file servers for whom the name servers maintain advertising and security information.
- Clients that mount resources from the RFS file servers.

You manage your RFS domain as an entity within your network administrative domain. Depending on the needs of your site, your RFS domain might include only certain machines on the network. Or, it might include all machines on your network, particularly if your network is small. Finally, a large network administrative domain might have several RFS domains, each with a primary name server maintaining different master files and supporting different file servers. In this last case, you can set up the primary name servers to share information among their serviced domains.

Figure 18-2 shows a large administrative domain, *arts.com*, that has two RFS domains, *rfsdance* and *rfsing*, and also runs NIS. The NIS domain is *arts.com*, and it maps service the entire network. The lines with arrows show how machines on the network interact with one another. Notice how machines that provide a such service as RFS, NFS, or NIS, can also be clients of other servers. Also notice how the NFS and NIS servers are included in the RFS domain because they are clients of RFS file servers.

Because all diskless clients must receive their *root*, *swap*, and */usr* files from an NFS file server, the figure does not show this relationship.

Figure 18-2 *Layout of Network Administrative Domain arts.com*



Your Role as an RFS System Administrator

Before you read further, consider which RFS functions your machine will perform. Will it be an RFS name server, file server or client? The remainder of the chapter contains procedures and theoretical material, grouped according to the type of RFS machine administered. The text below explains which materials you should read, grouped according to RFS machine type.

For the RFS Name Server

If you will administer an RFS name server, you are-in effect-the RFS domain administrator. Therefore, you should do the following.

- Read the next section, *Installing RFS*, and load the RFS software.
- Find out which machines on your network will use RFS before setting up the RFS domain. Discuss whether they will be RFS servers or clients with the network administrator, if you do not have this responsibility.
- Read this entire chapter.
- Use the procedures in *Setting Up the RFS Name Server*.
- Use the procedures in *Maintaining RFS Services*.

For an RFS File Server

If you will administer an RFS file server, you should use procedures in the following sections.

- *Installing RFS*, to load the RFS software.
- *Setting Up an RFS File Server*
- “Advertising Files”
- *Setting Up User and Group Security*

You should also familiarize yourself with *Mounting Resources*, which explains how clients mount your resources.

For an RFS Client

If your machine is to be an RFS client, you should read the next section, *Installing RFS*. Determine whether or not you need to load the RFS software.

Once RFS is installed on your machine, read the section *Mounting Resources* for instructions on accessing RFS resources.

18.2. Installing RFS

To use RFS, a machine must mount the software either from a local disk or from a remote NFS file server with RFS already loaded. You can install RFS software in either of two ways.

- By selecting it as an optional software package when you run SunInstall to install Release 4.1. (See *Installing SunOS 4.1* for complete information on running SunInstall.)
- By running `add_services` on a machine that has already loaded Release 4.1. (Refer to the chapter *Administering Workstations*, for complete instructions for running `add_services`.)

Installing RFS on a Machine with a Disk

The next instructions explain how to load the RFS software while installing Release 4.1 on a machine with a local disk, such as these types.

- A networked standalone that will support RFS activities, either as a client or server.

- An NFS file server that will export RFS software to its clients.
- A server machine that will serve both NFS and RFS to clients.
- An NFS file server that will be an RFS client.

On a machine with a local disk, insert your Release 4.1 tape and boot your machine according to the instructions in *Installing SunOS 4.1*. Then do the following.

1. Run SunInstall.
2. Fill out the Host Form with the appropriate information for your machine type.
3. Fill out the Disk Form as explained in *Installing SunOS 4.1*.
4. Fill out the Software Form and select “Own Choice.”
5. Type **y** for all software choices your configuration needs, *including RFS*.
6. If your machine is an NFS client server, access the Client Form, and specify your clients. All these machines then automatically receive RFS software when they mount `/usr` from your machine.
7. Finish running SunInstall, and reconfigure your kernel according to the instructions in the chapter *Reconfiguring the System Kernel*. Make sure the following lines are present in the configuration file and that they are not commented out:

```
options      RFS
options      VFSSTATS
pseudo-device tim64
pseudo-device tirw64
pseudo-device tcptli32
pseudo-device sp
pseudo-device clone
pseudo-device snit
pseudo-device pf
pseudo-device nbuf
```

8. Make the kernel file, then reboot your machine. Reconfigure the kernels of the diskless clients. They also must have the lines above enabled.

Loading RFS from an NFS File Server to Its Clients

Diskless and dataless clients of an NFS file server with RFS installed can run RFS software after a few adjustments are made. The next instructions explain how to set up NFS client machines to run RFS. They assume that you administer the NFS file server and have ultimate responsibility for setting up diskless and dataless clients. If you administer the NFS file server but are not responsible for clients, have the client administrators perform the instructions that pertain to their machines.

1. Become superuser on the NFS server.
2. Edit the kernels for all client machines to run RFS. Include the options and pseudo-device lines for RFS support, as shown in step 7 above.
3. Reconfigure the new NFS client kernels, as described in *Reconfiguring the System Kernel*.
4. Reboot the NFS client machines to have these changes take affect.

Note that although these machines are NFS clients, they can function as RFS file servers. The diskless client can share as RFS resources any directory hierarchies it mounts from NFS file servers. The dataless client can share as RFS resources any directory hierarchies mounted from NFS file servers, plus directory hierarchies on its local disk, including device files for the disk.

18.3. Setting Up the RFS Domain

This section explains how to set up your RFS domain after RFS software is installed. The following are general procedures for setting up the domain.

1. Make sure that all machines to use RFS either mount the software from local disks or from remote NFS servers.
2. Set up the primary RFS name server, as explained in the next subsection.
3. Copy the `/usr/nserve/rfmaster` file on the primary name server to all secondary name servers, RFS file servers, and clients in the new domain.
4. Initialize RFS on all machines in the RFS domain by running `dorfs init`.
5. Copy `/usr/nserve/auth.info/domain/passwd` to secondary name servers if your domain will use RFS passwords.
6. Start RFS on the primary name server.
7. Start RFS on any secondary name servers.
8. Start RFS on the file servers, optionally enabling user and group password verification.
9. Set up advertise tables on the file servers.
10. Start RFS on client machines, optionally defining passwords for these clients on the primary name server.

The remaining subsections fully describe these procedures.

Setting Up the RFS Name Server

The first step in setting up an RFS domain is to set up the primary RFS name server. You do all RFS domain administration from this server, including these functions.

- Setting up the `rfmaster` file, which defines the names and IP addresses of the primary and secondary name servers, including name servers for other accessible RFS domains.
- Maintaining password information for all machines in the RFS domain.
- Adding and removing machines from the RFS domain.

- Starting and stopping RFS for maintenance or in case of a crash.
- Maintaining a list of all machines in the RFS domain.

Before you start up RFS, you should have already determined which machines in your network will be RFS file servers and clients. Also, you should determine whether or not your RFS domain needs a secondary name server. Choose a secondary name server if your site requires a backup name server in case the primary goes down.

Defining the RFS Domain Name

After deciding the initial members of your RFS domain, you need to give it a name and define its primary and secondary name servers. Your RFS domain name can contain a maximum of 14 characters—any combination of letters (upper- or lowercase), digits, hyphens, and underscores. However, the name must be unique among the RFS domains in your network administrative domain. You will use this domain name shortly, when you create an `rfmaster` file and start up RFS for the first time.

Setting Up the `rfmaster` File

The `rfmaster` file contains the names and IP addresses of the primary and secondary RFS name servers. You need to create it for your domain before starting RFS. Here is a sample `rfmaster` for RFS domain `rfsdance`.

```
rfsdance      P      rfsdance.tap
rfsdance.tap  A      \x000214508190320d0000000000000000
rfsdance      S      rfsdance.tango
rfsdance.tango A      \x000214508190341c0000000000000000
```

Each name server is defined by at least two lines in the `rfmaster`. You define the server type, primary or secondary, on the first line, then give its IP address on the second line.

The line defining name server type has this form.

```
RFS_domain server_type RFS_hostname
```

The field *RFS_domain* represents the domain name you selected. In the example `rfmaster` above, this is `rfsdance`. For *server_type*, you enter either **P** for primary or **S** for secondary.

For *RFS_hostname*, you enter the full RFS hostname of the name server. This consists of the RFS domain name, a period, then the machine name of the name server, as in `rfsdance.tap` and `rfsdance.tango` above.

The second line, which contains the address of each name server, has this form.

```
RFS_hostname A hex_IP_address
```

The *RFS_hostname* field contains the full RFS hostname of the name server whose address you are about to supply. The **A** simply indicates that a network address is to follow.

In the field *hex_IP_address*, you have to supply the IP address of the name server in hexadecimal notation. Name server *rfsdance.tap* has the hexadecimal address `\x000214508190320d0000000000000000`. When you define the address of your name server, you must begin with the following characters.

```
\x00021450
```

These characters define the Internet address family (0002) and port number. The default port number used by Release 4.1 for RFS is 1450. However, this is not an official port number. Before using 1450, you should consult `/etc/services` to make sure it isn't in use at your site. If 1450 is in use, designate an unused port number after 0002. The remaining part of the address is the IP address of the server converted to hexadecimal representation.

You can use the `hostrfs` command to convert IP addresses to a form suitable for the `rfmaster` file. Here is its syntax.

```
% hostrfs machine_name [port_number]
```

where *machine_name* is the host name of your machine, as it appears in the `hosts` database. You can optionally specify *port_number*. Otherwise, `hostrfs` uses the default port number 1450.

For example, to find the `rfmaster` entry for a machine called *blues*, listed in `/etc/hosts` with an IP address of 137.50.44.88, type:

```
% hostrfs blues
```

You get the response:

```
\x0002145089322c580000000000000000
```

which is the format required by the `rfmaster` file.

In the `rfmaster`, you must separate fields on each line by a space or one tab. If a line becomes too long, you can extend it in standard SunOS fashion, by typing a backslash followed by a `RETURN`.

To communicate with other RFS domains on your network, find out the names and IP addresses of those domains' name servers. Then add this information to `rfmaster`. For example, suppose you want RFS domain *rfsdance* to communicate with domain *rfsing*. You would add the address of *rfsing*'s name servers to the `rfmaster` on *rfsdance.tap*, as follows.

```

rfsdance      P      rfsdance.tap
rfsdance.tap  A      \x000214508190320d0000000000000000
rfsdance      S      rfsdance.tango
rfsdance.tango A      \x000214508190341c0000000000000000

rfsing        P      rfsing.blues
rfsing.blues  A      \x0002145089322c580000000000000000
rfsing        S      rfsing.opera
rfsing.opera  A      \x00021450893232c37000000000000000

```

Follow the next procedures to create the `rfmaster`.

1. Become superuser on the prospective primary RFS name server.
2. Go to the directory `/usr/nserve`. (Note that `/usr/nserve` is actually a symbolic link to the `/etc/nserve` directory.)
3. Create the file `rfmaster` using your preferred text editor.
4. Change permissions on the newly-created file so that everyone can read it but only root can write it:

```
# chmod 644 rfmaster
```

Running `dorfs`

The next step in setting up the RFS name server is to initialize RFS, then start RFS services. You do this by running the command `/usr/bin/dorfs`. `dorfs` is very easy to use, as shown in the next steps.

1. Become superuser on the RFS name server.
2. Initialize RFS by typing this command.

```
# dorfs init domain tcp [port_num]
```

Replace the *domain* argument with the RFS domain name you selected. `tcp` represents the default SunOS network protocol type, TCP/IP. Replace *port_num* with the port number your site uses for RFS, if it does not use the default 1450.

After initializing RFS, you don't have to run `dorfs init` unless you want to change the RFS domain name. The `dorfs(8)` man page has more details about this issue.

3. Start RFS by typing the following command.

```
# dorfs start
```

After you have initialized RFS on the name server, you can edit the `/etc/rc` file to have RFS start automatically during the boot process. See the later subsection, *Starting RFS at Boot Time* instructions.

RFS Files and Daemons

`dorfs`' major function is to set up the RFS environment. It starts the `listener` daemon, which continually listens for RFS requests, much in the same way that `inetd` listens for network requests. `dorfs` also starts up the daemons `rfudaemon`, `rfs:server`, `rfs:recovery`, and `rfs:rfdaemon`.

`dorfs` also sets up files in the `/usr/nserve` directory hierarchy. Most of the files you need to use RFS reside in `/usr/nserve` and `/var/net`. Here are some files you'll use from `/usr/nserve`.

<code>auth.info/ domain</code>	<code>netspec nserve@</code>	<code>rfmaster loc.passwd</code>
------------------------------------	----------------------------------	--------------------------------------

<code>auth.info</code>	This directory contains RFS domain-wide access information for users and groups. All RFS passwords for a domain reside in this directory on the primary name server. The section, <i>Setting Up User and Group Security</i> , explains the files in <code>auth.info</code> in more detail.
<code>domain</code>	This file contains the RFS domain name.
<code>netspec</code>	This file simply contains the network protocol type (<code>tcp</code>).
<code>nserve</code>	This is a symbolic link to the <code>/usr/etc/nserve</code> command.
<code>rfmaster</code>	This file lists the names and addresses of your domain's primary and secondary name servers.
<code>loc.passwd</code>	This local file appears in <code>/usr/nserve</code> if you set up an RFS password for the local machine.

Setting Up a Secondary RFS Name Server

You may want to set up one or more secondary RFS name servers, depending on the size of your RFS domain. Secondary name servers maintain copies of the primary server's master files. That way, if the primary goes down, the secondary can take over providing the RFS domain with its needs. However, you cannot perform major maintenance on the secondary server.

Setting Up Secondaries for a New RFS Domain

If you are setting up a secondary name server for a brand new RFS domain, you use procedures that are similar to those for the primary. The steps below assume that you administer both primary and secondary name servers, and know the superuser password for both. If these machines have different administrators at your site, develop a communications method between yourselves, so that both primary and secondary servers start up within the appropriate time frame.

1. Make sure the RFS software is available on the secondary, either from a local disk or NFS file server.
2. Become superuser on the primary name server, edit the `rfmaster`, and add the appropriate entries for the secondary server.

- Copy the `rfmaster` file from the primary name server to `/usr/nserve` on the secondary name server(s).

```
% rcp /etc/nserve/rfmaster secondary:/usr/nserve
```

- From a window on your primary name server screen, remotely access the secondary server.
- Become superuser on the secondary and initialize RFS with this command.

```
# dorfs init domain tcp [port_num]
```

Replace the *domain* argument with your RFS domain name. Replace *port_num* with the port number your site uses for RFS, if it does not use the default 1450. Note that you only have to use `dorfs init` once, when you start up RFS on a machine for the first time.

- In the window where you are superuser on the *primary* name server, start RFS by typing this.

```
# dorfs start
```

- In the window where you are superuser on the *secondary* name server, start up RFS by typing:

```
# dorfs start
```

Note: From the time you run `dorfs start` on the primary, you have two minutes to run `dorfs start` on the first secondary server. If you do not keep within this time frame, server start up will fail.

- Become superuser on any additional secondary name servers you may have at your site, and run `dorfs start`.
- Have the administrators or users, if applicable, run `dorfs start` on all other machines in the RFS domain.

Setting Up Secondaries for an Existing RFS Domain

If you are adding the secondary name server to an existing RFS domain, do the following:

- Become superuser on the primary name server.
- Update the `rfmaster` file with the proper entry for the new secondary server, as explained in the previous subsection.
- Run

```
# dorfs stop
```

on all machines in the RFS domain. (See the subsection, *Maintaining RFS Services*, for more information about `dorfs stop`.)

- Run `dorfs start` on the primary.

5. Follow steps 4 through 9 in the previous subsection, *Setting Up Secondaries for a New RFS Domain*.

If user and group security is in effect for your RFS domain, you must retain a copy of the files in the directory `auth.info` on each secondary name server. However, you cannot add or remove hosts or change host passwords on a secondary, even when it takes over for a failed primary name server. You can only pass name server responsibility back to the primary when it comes up.

Though a primary name server can serve several RFS domains, secondary servers can only serve one RFS domain. This simplifies the process of returning name server responsibilities to a primary after it comes back up.

Setting Up an RFS File Server

As administrator of an RFS file server, your primary task is advertising your machine's resources and, optionally, setting up security features to protect these resources. Yet before you can tell the domain about your resources, you must set up your machine as an RFS file server. Later sections explain how to advertise resources and set up optional security measures.

Before you start up RFS on your file server, you do the following.

- Know the name of the RFS domain you want to join.
- Ensure that the primary name server for the domain has been started.
- Ensure that your file server mounts RFS software, either from a local disk or from a remote NFS file server.

Note that your RFS file server does not necessarily have to advertise resources on its local disk. In fact, it does not need a local disk. It can simply mount a directory hierarchy from an NFS file server, then advertise this hierarchy as an RFS resource. Moreover, if your site doesn't have many machines, it is possible to set up a machine to function as both RFS file and name server.

Procedures for Starting RFS on the File Server

Here are the procedures for starting RFS on your file server.

1. Copy the `rfmaster` file from the primary name server's `/usr/nserve` directory into your file server's `/usr/nserve`.

```
% cd /usr/nserve
% rcp name_server:/etc/nserve/rfmaster .
```

2. Become superuser and change permissions for your machine's `rfmaster` file, so that everyone can read it, but only the superuser can write to it:

```
# chmod 644 rfmaster
```

3. Initialize RFS on your file server by typing this command.

```
# dorfs init domain tcp [port_num]
```

Replace the *domain* argument with your RFS domain name. Replace

port_num with the port number your site uses for RFS, if it does not use the default 1450. Note that you only have to use `dorfs init` once, when you start up RFS on a machine for the first time.

4. Type the following to set up basic RFS user and group mapping rules files.

```
# auth.info
# cat > uid.rules
global
default transparent
^D
# cp uid.rules gid.rules
```

Note that these rules files must be present when you run `dorfs start`. They basically allow transparent mapping of user and group IDs. To implement more restrictive security measures, refer to the later section, *Setting Up RFS Security*.

5. Start up RFS by typing the following.

```
# dorfs start
```

Note that `dorfs start` includes a `-v` option, enabling RFS password verification for client machines that want to use your file server's resources. The subsection, *Setting Up User and Group Security*, explains how RFS password verification works. If you want to set up RFS password verification, refer to this section before you start up RFS.

After you complete these steps, you probably will want to have the file server automatically start RFS when it boots, just as it starts NFS. The later subsection, *Starting RFS at Boot Time*, contains instructions for doing this. Thereafter, your name server will automatically start RFS, look for the existence of the `/etc/rstab` file, and advertise any resources listed in this file.

After you finish setting up the RFS file server, you should determine which resources it will advertise and set up its advertise tables. The section, *RFS File Servers and Resource Sharing*, explains resource advertising.

Starting RFS on a Client

You can start up RFS on any type of networked Sun computer that you want to make an RFS client. But first you must do the following.

- Know the name of the RFS domain you want to join.
- Ensure that the primary name server for the domain has been started.
- Let the administrator of the primary name server know that you are about to copy its RFS files, so that file permissions can be adjusted if necessary. Confirm with the administrator of the primary name server that it has an entry for your machine in its `rhosts` file.
- Ensure that your machine has RFS software available, either from a local disk or by mounting the `/export/root`, `/usr` or `/export/exec/kvm` directory of a remote NFS file server.

Then follow these procedures on your client machine.

1. Copy the `rfmaster` file from the primary name server.

```
% cd /usr/nserve
% rcp name_server:/etc/nserve/rfmaster .
```

2. Initialize RFS.

```
# dorfs init domain_name tcp
```

Note that you only have to do steps 1 and 2 when you start up RFS on a machine for the first time. Thereafter, you only need to perform the next step to start RFS.

3. Start RFS.

```
# dorfs start
rfstart: Please enter machine password
```

4. Type an RFS password for your client machine. If password verification is not used in your RFS domain, simply press **RETURN** in response to the prompt.

If password verification is in place in your RFS domain, the name server administrator may have already asked you to type an RFS password on the name server. Type that password now. Or, create a new one now, if this is what the name server's administrator has asked you to do. The RFS password must be at least six characters long and contain at least one numeric character.

When you finish setting up the RFS client, go to the section, *Mounting Resources*, for instructions for accessing the available resources in the RFS domain.

Starting RFS at Boot Time

After you initialize and start up RFS, you can have your machine automatically start RFS when it boots. In Release 4.1, the startup script `/etc/rc` contains the following lines:

```
#if [ -f /usr/nserve/rfmaster ]; then
#   echo -n starting rfs:
#   if [ ! -f /usr/nserve/loc.passwd ]; then
#       echo "" > /usr/nserve/loc.passwd
#       echo "" > /usr/nserve/loc.passwd.dummy
#   fi
#   /usr/bin/dorfs start; echo ' done.'
#   if [ -f /usr/nserve/loc.passwd.dummy ]; then
#       rm -f /usr/nserve/loc.passwd /usr/nserve/loc.passwd.dummy
#   fi
#fi
```

These lines enable RFS services. Simply become superuser on your machine, edit `/etc/rc`, and remove the comments (`#`). The next time your machine boots it

will automatically start RFS.

18.4. Advertising Resources from an RFS File Server

This section describes how to use RFS to share, or *advertise*, resources with other machines in an RFS domain. Below is a list of generic instructions for setting up advertising on an RFS file server.

1. Determine which resources you want to share, and which RFS clients you will allow to access them.
2. Determine if your file server will use RFS security measures, such as RFS password verification and user and group ID mapping in addition to the rules files created when you started RFS. Should you want this extra protection, go to the section, *Setting Up RFS Security*.
3. Create names for your resources.
4. Verify that the resource names are unique within the RFS domain by consulting the advertise tables.
5. Advertise the resources by using the `adv` command.
6. Optionally set up the `rstab` and `host.alias` files.
7. Maintain advertising services, including unadvertising certain resources.

The actions you take for steps 1 and 2 depend on your site's needs. The remainder of this section explains steps 3 through 7 in detail.

Naming Your Resources

You can advertise different types of directory hierarchies as resources. For example, you can advertise individual directory hierarchies, an entire `/dev` hierarchy of device files, or named pipes. Your RFS file server can mount these directory hierarchies from a local disk or mounted tape. Additionally, you can advertise as resources most directory hierarchies that your RFS file server mounts via NFS. In this way, a diskless NFS client can function as an RFS file server. Or, small sites can have one file server providing both NFS and RFS services.

You can advertise the `/export` and `/usr` file systems as RFS resources, but diskless and dataless clients cannot mount them using RFS until they finish the boot process. Diskless and dataless clients must mount these file systems from NFS in order to boot. You can share a single file system through both RFS and NFS services simultaneously.

Checking the Advertise Tables

When you advertise an entity as a resource, you must assign it a resource name that is unique within your RFS domain. If you are setting up advertising for a brand new RFS domain, this merely entails taking care to give resources different names. However, if your RFS domain already exists, you must verify that your resource name is unique by checking the advertise tables. These tables maintain up-to-date information about all advertised resources. Your RFS file server maintains a local advertise table; the RFS name server maintains an advertise table listing available resources for the RFS domain.

The `adv` command, when run without options, displays your local machine's advertise table. Here is an example:

```
% adv
ALLSING /home/singers "Singer's database" read-write rfsing
```

The first column of the display is the name of the resource. The remaining information is identical to that which you supply to the `adv` command, as described shortly.

Note that the fifth column optionally lists clients to whom you have restricted the resources. The values listed in this last field can be any of the following.

- Host names in the domain, such as `elvis` or `aretha` in domain `rfsing`.
- Host names in other domains, such as `rfsdance.raks` or `rfsdance.ballet`.
- Domain names, such as `rfsing`, as shown in the display above.
- Aliases listed in `/etc/host.alias`, which is described later in this section.
- `unrestricted` if the resource is not restricted to certain hosts.

Displaying the local advertise table is sufficient if your RFS file server is the only one in the RFS domain or the first file server set up. If other file servers are in the RFS domain, run `/bin/nsquery` instead.

`nsquery` without options displays the advertise table on the primary RFS name server for your domain.

RESOURCE	ACCESS	SERVER	DESCRIPTION
ARTS	read-write	rfsdance.jazz	Artist list
LOCAL_SUN4	read-only	rfsdance.dancers	/usr/local dir

The full syntax of `nsquery` follows.

```
nsquery [ -h ] [ name ]
```

Note: The output displayed by `nsquery` does not indicate whether you have permission to access the resource.

Using the `adv` Command

name can be an RFS domain name, an individual hostname, or a combination of the form, *domain_name.hostname*. Specifying the `-h` option results in a display without the column titles. (See `nsquery(8)` for more details.)

The `adv` command makes resources from your machine accessible to other machines running RFS software. You must become superuser in order to run `adv`. When you want to advertise a resource for the first time, use `adv` with the following syntax.

```
/bin/adv [-r] [-d description] resource pathname [clients...]
```

The `adv(8)` man page completely describes the command and its option. Here is a description of the most commonly used options, which are shown in the previous syntax statement.

- r** The **-r** option advertises the resource with read-only access. If you do not specify **-r**, the remote hosts will have read/write access to the resource.
- d *description*** The **-d** option indicates that the next argument, *description*, is a description of the resource. The description can have from zero to 32 characters. Enclose the description in quotation marks ("").
- resource*** This is the resource name you assign, which the file server and authorized clients use to identify the resource. The name is limited to 14 printable ASCII characters; you cannot use slash (/), period (.), or white space. Remember that the resource name must be unique within your RFS domain.
- pathname*** This is the local pathname of the directory hierarchy you want to share. Your RFS file server must already mount the directory, either locally or through NFS, and it cannot be already advertised.
- clients...*** This is an optional list of one or more host or domain names that you want to restrict this resource to. If you do not specify clients, an RFS host that can access your RFS file server can mount the resource.
- Specify clients by using the form *hostname* or *domain.hostname*. Separate each client name with a space. You can specify an entire domain by typing the domain name with a trailing period (*rfsing.* as an example). You can also define aliases so that a single client name can represent a group of hosts and/or domains. (See the subsection *Aliases* below.)

Now suppose you want to advertise a database package that your RFS file server ballet has on its local disk. Assume the full pathname of that package is `/home/mydir/special.db`, and that you will permit all RFS machines at your site to use it. You would run `adv` as follows.

```
# /bin/adv -d "Dancers database" DANCERSDB /home/mydir/special.db
```

After you run `adv`, your resource is registered with the RFS name server for your domain. Any network host that can reach your domain can find the resource listed in the display from `nsquery`.

RESOURCE	ACCESS	SERVER	DESCRIPTION
ARTS	read-write	rfsdance.jazz	Artist list
LOCAL_SUN4	read-only	rfsdance.dancers	/usr/local dir
DANCERSDB	read-write	rfsdance.ballet	Dancers database

All the host will know is the resource name, the short description you assigned,

that the resource resides on your RFS file server, and its access permissions.

When you want to modify the parameters of an existing resource, use either of the following forms of `adv`.

```
adv -m resource -d description [clients ...]
```

```
adv -m resource [-d description] clients ...
```

The `-m resource` option indicates that you are about to modify the *description* or *client* fields for an advertised resource listing. You cannot use it to change the read/write permissions of a resource.

For example, suppose you want to restrict the resource DANCERSDB only to client machines `raks`, `samba`, and `jazz`. You would become superuser on `ballet` and type the following:

```
# adv -m DANCERSDB raks samba jazz
```

Advertising Devices as Resources

A distinctive feature of RFS is the ability to advertise devices as resources. If your RFS file server has local devices attached to it, such as a disk or tape drive, you can share them with RFS clients. You accomplish this by advertising your file server's `/dev` directory as a resource. Thereafter, remote clients can access the contents of an entire disk or a tape in a drive by issuing one `mount` command, then running SunOS commands on the remote drive, as described in the subsection, *Mounting Remote Devices*.

Note that when you share a device as a resource, RFS clients will access it as a raw device. This means the information will be accessed in the form of bytes. This is fine, even appropriate, for tape devices, particularly if you want use them with commands like `tar`. However, this is not appropriate for devices that have file systems, as is the case for most disks. Therefore, you have to perform an explicit local mount on each partition of the disk before clients can access its file systems as RFS resources. If you specified these disks when you installed 4.1, SunInstall will have already created the local mounts for you. The chapter *Adding Hardware to Your System* contains instructions for explicitly mounting file systems on disks you may have added after installing the operating system.

Here are sample instructions for advertising devices on your RFS file server as resources.

1. Become superuser on your file server.
2. Create a subdirectory in `/dev` called `/rdev` as follows.

```
# cd /dev
# mkdir rdev
```


3. Link the devices that you wish to make available in the `rdev` directory.

```
# cd /dev/rdev
# ln /dev/rst8 rst8
# ln /dev/rmt8 rmt8
```

4. Advertise the `/dev/rdev` directory as a resource.

```
# adv -d "elvis devices" ELVIS_DEV /dev/rdev
```

The `rstab` File

The `/etc/rstab` file enables your RFS file server to automatically advertise its resources when it boots. The visible results are similar to those of `/etc/exports` in NFS. But unlike `exports`, `rstab` is actually a shell script that executes when you run `dorfs start`.

`SunInstall` does not create `/etc/rstab` when you select RFS as optional software; you have to create it yourself. The `rstab` file entries should consist of the entire command line for each advertised resource provided by the file server. Here is a sample `rstab` for server `rfsdance.dancers`:

```
#!/bin/sh

adv -r -d "/usr/local files" LOCAL_SUN4 /usr/local/sun4
adv -d "Home directories" HOME_DANCERS /home/dancers
```

To create the `/etc/rstab` file, do the following.

1. Become superuser and go to `/etc`.
2. Create a file called `rstab` using your preferred text editor.
3. Type the following on the first line to have the Bourne shell execute commands that follow.

```
#!/bin/sh
```

4. On the succeeding lines, type every `adv` command you used to advertise your current resources. (If you can't remember all your resources, use `adv` without options to display them.)
5. Exit `/etc/rstab`, then check its permissions by running `ls -l`.

```
# ls -l /etc/rstab
-rw-r--r-- 1 root          89 Sep  6 17:18 /etc/rstab
```

6. Change permissions as follows, so that everyone can read and execute the script.

```
# chmod 755 /etc/rstab
```

Aliases

Rather than specify a long list of clients to `adv`, you can define aliases for them. This is especially useful if your environment allows only certain individuals or machines to access particular resources.

You use the `/etc/host.alias` file to define these aliases. The `adv` command reads `/etc/host.alias` to find the definitions of any aliases in the `clients` field.

Each entry has the following format.

```
alias name client1 client2 client3 ...
```

`name` represents the character string you want for the list of clients. Each client can be a hostname, RFS domain name, or an alias name previously defined in the file. Separate the three fields by blanks or tabs. You can escape a new line character with a backslash if you have too many clients to fit on one line.

Here is an example `/etc/host.alias` file.

```
alias    dancers  samba ballet raks jazz tap afro hula
alias    singers  ella elvis aretha hank placido pato
```

The operating system does not create `/etc/host.alias`. Therefore, you must create it by hand. Become superuser, so that the root user ID automatically owns the file. Then create the file with your favorite text editor, using the syntax shown above. When you have finished, check the file protection mode given to `host.alias`. It should be 644. Use `chmod` to change it, if necessary.

Maintaining Resources

Once you have advertised your resources, you will have to perform maintenance activities on them from time to time. Most of this maintenance involves changing permissions on the resource or making a resource unavailable. This subsection describes the following resource maintenance activities:

- Changing resource security
- Monitoring resource usage
- Unadvertising resources

To completely remove a resource from the RFS file server, you have to force clients to unmount it. The next major section, *Mounting Resources*, contains procedures for unmounting resources.

Maintaining Resource Security

Once you advertise a resource, you can, if required, set up certain levels of RFS security to protect it. The section, *Setting Up RFS Security*, completely explains these security levels. The following are brief definitions of the levels:

Connect security	Only those remote hosts that pass your security checks can even connect to your host. You may have indicated that only those hosts you have a record of can connect.
Mount security	You may have advertised your resource so that only selected hosts can mount it (see the <i>client</i> option of the <i>adv</i> command). (See <i>rfstart -v</i> in <i>Setting Up RFS Security</i> .)
User security	Permissions that remote users have to your resources are set on a host-by-host basis. In other words, the user and group mappings you set up for a remote host will apply to any of your resources which that host mounts.
SunOS system security	Standard SunOS system access security, governing read, write, and execute permissions, applies to any advertised resource.

Monitoring Resource Usage

You can determine who is mounting your advertised resources by displaying the domain mount table. You do this by running the `rmntstat` command. Its syntax follows.

```
rmntstat [-h] [resource]
```

The *resource* argument displays mounting information for the specified resource. The `-h` option runs `rmntstat` without its column headers.

If you run `rmntstat` without options, here is the resulting display.

```
# rmntstat
RESOURCE      PATH          HOSTNAMES
ALLSING       /db/singers   elvis leontyne kulthum
```

Note: If `unknown` appears in the `pathname` field, it means you have unadvertised the resource, but it is still mounted on the listed remote machines.

You can use `rmntstat` to find out who is currently using a resource, in case you have to unadvertise it to change permissions, or possibly take down your RFS file server for maintenance. Moreover, you can use `rmntstat` to determine if any machines accessing the resources might not be authorized to mount them. In this way, you can evaluate security requirements for the particular resource.

RFS provides additional commands for monitoring resource usage, such as `fusage` and `fuser`, which are described later in the chapter.

Unadvertising Resources

The `unadv` command “unadvertises” any of your RFS file server’s resources. That is, it removes the resource from the advertise tables on your file server and the RFS name server.

You must use `unadv` under the following circumstances.

- Before unmounting one of your file systems that contains an advertised directory

- To restrict a previously-shared directory to only local access
- To change the pathname, resource name, or read/write permissions of an already advertised resource. In this case, you unadvertise the resource, then re-advertise it.

`unadv` does not remove a resource from any remote RFS client that currently mounts it. (You must use the `fmount` command to do this.) `unadv` does, however, prevent subsequent mounting attempts by remote hosts.

To unadvertise a resource on your local RFS file server, use the following syntax of `adv`.

```
unadv resource
```

You must be superuser to run `unadv`.

For example, suppose the directory hierarchy `/db/singers` on RFS file server `aretha` is advertised as resource `ALLSING` and is writable by everyone in the `rfsing` domain. Now suppose you must make `ALLSING` more secure and decide to change access to the resource to read-only. First, you type the following to unadvertise `ALLSING` to the entire domain.

```
# unadv ALLSING
```

Then you re-advertise `ALLSING` as follows.

```
# adv -r -d "Singer's database" ALLSING /db/singers rfsing
```

Any RFS clients who mounted `ALLSING` before you unadvertised it can still modify files in the `/db/singers` directory hierarchy. But clients who subsequently mount `ALLSING` can only read files in `/db/singers`.

Because you can set up `/etc/rstab` to automatically advertise resources during booting, you may have to remove any commands for resources that you want permanently unadvertised.

If you are an RFS domain administrator, you can use `unadv` to remove resources from an RFS file server in your domain. However, you should not do this unless the file server has gone down. Refer to the later section, *Domain Maintenance*, for more information.

18.5. Mounting Resources

Any machine that has started RFS software can use the `/usr/etc/mount` command to mount an advertised resource from an RFS file server. The machine that mounts the resource is considered an RFS client. You perform RFS mounts using steps similar to those for NFS mounts, as the following summary indicates:

1. Run `nsquery` to display the list of resources available to your RFS domain. (Refer to the previous section, *Checking the Advertise Tables* if you are unfamiliar with the `nsquery` command.)
2. Choose a directory to serve as a mount point for your selected resources. The mount point can be an existing, empty directory, or you can use `mkdir`

to create a new mount point directory.

3. Become superuser on your local machine.
4. Use the `mount` command with the `-d` option to mount the resource.

Your machine then sends a request to the RFS file server that advertised the resource. If you have permission to mount the resource, it will be connected to the mount point that you specified.

Using `mount` to Mount RFS Resources

Use `mount` with the following syntax to mount an RFS resource.

```
mount [-r] -d resource mountpoint
```

The options are as follows.

<code>-r</code>	Specifies that the resource should be mounted read-only. If you do not use <code>-r</code> , the resource is mounted with read/write permissions. You can only mount a remote resource with read/write permissions if it is advertised that way. If the resource is advertised with read-only permissions, you must use <code>mount</code> with the <code>-r</code> option; otherwise the mount will fail. Refer to the advertise table displayed with <code>nsquery</code> for the permissions applied to your resource.
<code>-d resource</code>	The <code>-d</code> option indicates that you are going to mount an RFS resource. <i>resource</i> gives the resource name, as shown by <code>nsquery</code> .
<i>mountpoint</i>	Specifies the mount point directory where your machine is to connect to the remote resource.

Now, suppose you just started RFS on your Sun-4 machine and want to mount some resources available to your RFS domain. You run `nsquery` and receive the following display.

```
LOCAL_SUN4 read-only rfsdance.dancers /usr/local dir
```

You then use an empty mount point, `mnt` for example, and issue the following to connect to this resource:

```
# /usr/etc/mount -r -d LOCAL_SUN4 /mnt
```

Note the use of the `-r` option because `LOCAL_SUN4` is advertised as read-only.

Once you have mounted RFS resources, they appear in displays from `df` and `mount` without options. Here is sample output from `df` with RFS resources mounted:

```
# df
Filesystem      kbytes   used   avail capacity  Mounted on
/dev/sd0a        7431    4577   2110    68%      /
/dev/sd0g       65951   41224  18131   69%      /usr
.
.
dancers:/home/dancers
855329 281124 488672   37%      /home/dancers
raks:/usr/man    89325   53637  26755   67%      /share/man
.
.
LOCAL_SUN4      112471 105272   7199   81%      /mnt/local
```

Note how the display shows 4.2, NFS, and RFS mounts.

A display from `mount` with no options could resemble the following.

```
# /usr/etc/mount
/dev/sd0a on / type 4.2 (rw)
/dev/sd0g on /usr type 4.2 (rw)
.
.
dancers:/home/dancers on /home/dancers type nfs (rw,hard)
raks:/usr/man on /share/man type nfs (ro)
.
.
LOCAL_SUN4 on /mnt/local type rfs (ro)
```

Note how `rfs` is listed after `type`, as are `4.2` and `nfs`. The display shown by `mount` is sometimes referred to as the *host mount table*.

Mounting Resources During Booting

You can add RFS resource entries to the `/etc/fstab` file, so that your machine automatically performs RFS mounts during booting. (The chapter *The Sun Network File System Service* describes `/etc/fstab` from the NFS perspective.)

The format for RFS entries in the `fstab` file is similar to those used for NFS.

```
resource mountpoint type options freq pass
```

The variables you supply are defined below and in the `fstab(5)` man page.

<code>resource</code>	Name of the resource you want to mount, as it appears in the advertise tables displayed by <code>adv</code> and <code>nsquery</code> .
<code>mountpoint</code>	Pathname of the local directory where you want to mount the resource.
<code>type</code>	Type of mount you want to perform; in this case, you specify "rfs."

options	You can specify three options for type <code>rfs</code> . The first option, <code>ro rw</code> , assigns either read-only or read-write permission to the resource. They are <code>bg fg</code> and <code>retry=n</code> . If the first attempt to mount the resource fails, the <code>bg</code> option tells <code>mount</code> to try again in the background. If you give <code>mount</code> the <code>fg</code> option, it retries the mount in the foreground. The <code>retry=n</code> option lets you substitute for variable <code>n</code> the number of times your machine should retry to mount the resource before giving up.
freq	Interval in days between dumps of the resource. This must always be zero for an RFS mount.
pass	<code>fsck</code> pass in which the partition containing the resource is checked. This must always be zero for an RFS mount.

To add entries to `/etc/fstab`, do the following.

1. Run `nsquery` and decide which advertised resources your machine should always mount when it boots.
2. Become superuser on your machine.
3. Determine the mount points where you want to attach these resources. Either use existing empty directories, or make new mount point directories as follows.

```
# mkdir mount_point
```

4. Edit `/etc/fstab` and add entries for all RFS resources you need, following the syntax shown previously.
5. Close `/etc/fstab`. Unmount all directories in the `fstab` file, then remount them, as follows:

```
# umount -a
  The system will display messages if it cannot unmount a particular directory

# mount -a
```

After you have finished these instructions, your machine can access the listed RFS resources, as well as any NFS file hierarchies listed in `/etc/fstab`.

Mounting Limitations for RFS Clients and File Servers

Mounting resources through RFS has ground rules that you need to observe. The most basic is: resources advertised with read-only permissions must be mounted read-only. If a resource is advertised as read/write, you can mount it read-only or read/write.

Other circumstances may arise where mounting has limitations. Follow the suggestions listed below to avoid unexpected results.

1. Mounting over system and other crucial directory hierarchies.

Do not use as mount points any directory hierarchies containing crucial files that define the environment for your local machine. This will result in these files becoming inaccessible to your system.

For example, you shouldn't mount a remote `/dev` on your machine's `/dev` directory, or you will make your machine's console (`/dev/console`) inaccessible. As another example, if you mount a remote `/etc` directory on your `/etc` directory, you cover your local `hosts`, `passwd`, `fstab` and `nserve` files, to name a few.

Some other directories that fall into this category are: `root (/)`, `/usr`, `/usr/bin`, `/usr/nserve`, `/usr/net`, and `/shlib`. You can, of course, mount these directories in other directories on your computer with no ill effects.

2. Mounting spool and work directories

Avoid mounting spooling or workspace directories from an RFS file server onto the same directory on another machine. Applications such as `uucp` and `lp` can run into problems when multiple machines are trying to create spool files or lock files in the same directory. For example, if you share the `/var/spool/locks` directory, by using a `tty` device for `uucp` on one machine, you would prevent use of a device of the same name on another machine. Also, mounting `/tmp` can cause collisions among temporary files.

Note: You can mount these directories in principle, but be aware that the symbolic links in them will not work.

Mounting Remote Devices

Some RFS file servers will advertise their remote devices as resources. This feature enables you to mount a remote tape or an entire disk—depending, of course, on how the file server has advertised the resource.

Suppose you run `nsquery` and see a resource called `ELVIS_DEV`, described as Elvis' devices. Here are instructions for mounting the available device files.

1. Become superuser on your RFS client.
2. Make a mount point, such as `rdev` below, as a subdirectory of `/dev`.

```
# mkdir /dev/rdev
```

3. Mount the resource on the newly created mount point.

```
# mount -d resource /dev/rdev
```

resource is the resource name, such as `ELVIS_DEV`.

4. To find out which devices you have mounted, type the following.

```
# ls -l /dev/rdev
crw-rw-rw- 2 root staff 18,  8 Oct 31 15:58 /dev/rst8
crw-rw-rw- 2 root staff 18,  8 Oct 31 15:58 /dev/rmt8
```

Note that in this example, the administrator of the RFS file server has only made tape devices available as resources. To access the mounted tape

devices, use `tar` and `dump` with the `-f` option.

Mounting Guidelines for RFS File Servers

If your machine is an RFS file server, you should additionally follow these mounting guidelines.

- You can mount a local file system on the resource directory or its subdirectories. The new file system becomes part of the advertised resource.
- You cannot mount a remote resource directly on an advertised directory.
- You can mount a resource from another RFS file server on subdirectories of your advertised resource. However, the remote resource you mount will not become part of the resource you advertise. Only local users will be able to access it. (Remote users will be able to see this mount point directory, but will get a “multihop attempted” error message if they try to run `ls -l` on the directory.)

Unmounting Resources

You can unmount any remote resource you have mounted by using the `umount` command.

Unmounting Resources on an RFS Client

There are two generic steps for unmounting a resource your client machine mounts from an RFS server.

1. Make sure that no one else is using that resource.
2. Run the `umount` command to remove the resource.

Before running `umount` on your RFS client, make sure that no remote machine or attached terminal is using the resource in question. You can find out who is using the resource by running the `fuser` command. `fuser` lists the processes on your machine that are accessing a mounted resource or directory hierarchy. If you wish, you can also use `fuser` to kill all processes relating to the specified resource.

You must be superuser to run `fuser`. Use the following syntax for reporting on remote resources:

```
fuser [-ku] resource ...
```

The `-u` option produces a longer report of processes that have files open in any directory or subdirectory relating to the resource. The `-k` option kills all processes relating to the resource. (Refer to the `fuser(8)` man page for more information about this command.)

For example, assume you have a machine that mounts the resource `LOCAL_SUN3` from server `dancers`. `dancers`' administrator has requested you to unmount `LOCAL_SUN3` while it is being backed up. To find out if any terminal users (or any remotely logged in workstations) are accessing `LOCAL_SUN3`, type the following.

```
# /usr/etc/fuser -u LOCAL_SUN3
```

Once you have ensured that no one is using the mounted resource, you use the following syntax of `umount` to unmount it.

```
/usr/etc/umount -d resource
```

The `-d` option tells `umount` to perform an RFS unmount on *resource*—the name of the resource you want to unmount.

For example, you would use the following on the machine above to unmount the resource.

```
# /usr/etc/umount -d LOCAL_SUN3
```

Unmounting Advertised Resources on an RFS File Server

The next instructions pertain to administrators of RFS file servers. Occasions will arise when you have to unmount an advertised resource that your file server mounts from a local device. You cannot use `umount` in this instance if any part of the local file system is mounted by remote clients. In most cases, you simply ask users mounting the resource to unmount it. This enables them to remove the resource in an orderly fashion.

However, emergency situations may arise when you have to unmount one of your advertised resources immediately. Should this happen, you can use the `fumount` command to unadvertise your resource and forcibly unmount it on all clients. Only use `fumount` in urgent cases where the resource must be removed, because you may cut off remote processes that access the resource.

You must be superuser to run `fumount`. Its syntax follows.

```
fumount [-w sec] resource
```

where the `-w` option tells `fumount` to wait *sec* seconds before unmounting *resource* from the clients.

The following occurs when you execute `fumount`.

1. Your RFS file server unadvertises the resource.
2. If you ran `fumount` with a grace period of several seconds, the following shell script runs on all RFS clients currently accessing the resource:

```
/usr/bin/rfuadmin fumount resource
```

The `rfuadmin` shell script runs whenever unexpected events occur in the RFS environment. It is fully described in the `rfuadmin(8)` man page. By default, `rfuadmin` writes this message to all terminals.

```
resource is being removed from the system in ## seconds.
```

You can edit `/usr/bin/rfuadmin` to tailor the action taken in response to `fumount`.

3. When the client machine gets this message, it tries to kill all processes that are using RFS files taken from the resource.
4. After the grace period of *sec* seconds, the resource is removed from all RFS clients that mount it.
5. By default, `rfuadmin` will start up again on each RFS client to send this message to all terminals:

```
resource has been disconnected from the system.
```

6. `rfuadmin` then tries to remount the resource, using the `bg` option of `mount` until it succeeds.

18.6. Setting Up RFS Security

Should your machine or RFS domain require tight security controls, you can set up RFS security measures in addition to the SunOS security features discussed earlier in the chapter *Administering Security*. This section explains how to implement RFS security features. It begins with an overview of security levels from the RFS perspective. The remaining subsections explain how to implement these features on RFS file servers, clients, and name servers.

RFS Security Levels—an Overview

RFS security exists on three levels: connect, mount, and user/group.

Connect Security

From an RFS perspective, *connect security* is a resource's first line of defense, in that it lets you define which remote RFS clients are allowed to connect to a particular machine. When a remote host attempts to mount a resource advertised by an RFS file server, it tries to set up a virtual circuit across the network to the file server. Once this virtual circuit is set up, the host can mount any resource the file server makes available to it. The circuit is closed when the last resource is unmounted.

Before this virtual circuit is created, the RFS client must pass the connect security checks set up on the RFS file server. The checks can be either of the following:

- Starting RFS on the file server by running `dorfs start -v`.

The `-v` option checks the RFS password of each client machine that tries to mount the server's resources. If a client machine has an RFS password in the file server's `domain/password` file, it must supply the correct password to the file server in order to connect. (The `domain/password` file is described shortly.) Machines that supply the wrong RFS password or that do not have a password at all cannot connect to the RFS file server.

- Running `dorfs start` on the RFS file server without specifying the `-v` option.

In this case, when a client tries to mount the RFS file server's resources, the server still checks its local `domain/password` file for an RFS password for that client. If the server finds an entry in the file for the client, the client must provide the correct RFS password in order to connect. If the server does not find an RFS password for the client, it allows the client to connect.

RFS connect security and how it applies to each type of RFS machine is discussed shortly.

Mount Security

Mount security is the second level of RFS security. It enables you to specify which resources a remote host can mount, once it has permission to access an RFS file server. How the remote machine mounts these resources depends on how you advertised each resource, as in these examples.

- Any host can mount

You can advertise a resource so that any host with permission to access your RFS file server can mount the resources.

- Some hosts can mount

By using the *clients* option of `adv`, you can restrict access to the resource to certain machines. Therefore, any remote host trying to mount the resource must be one of these client machines.

- Hosts can mount the resource, but with read-only permission.

By using `adv` with the `-r` option, you restrict remote hosts to mounting your resource read-only, rather than with the default read-write permission.

The previous section, *Mounting Resources*, deals with mount security in detail.

User and Group Security

User and group security is the third level of RFS security that you can establish for a mounted resource. You can define remote users' permissions, which are then mapped into your RFS file server's user and group list. This sets the permissions they will have to your resources. Thereafter, when remote users mount your resources, they do so with user and group permissions according to the *mapping rules* you have defined.

You define mapping rules in rules files that apply on a global or per-machine basis. The global rules set user and group permissions for all machines that do not have explicit host mapping rules. When you start RFS on a file server, you have to define a default set of rules files for user and group.

Here is how you can map users on remote hosts into your machine. These rules apply to both global and host mapping.

- No mapping

If you do not set any special mapping for any remote machine, all remote users are mapped into your machine as a *special guest* user ID/group ID. This is the easiest approach because you don't need to keep any records for the remote host.

- Default mapping

Here you set up mapping rules so that all remote users map into one of these permissions:

- Local user ID number that matches each remote user's ID (default transparent)

- Single local ID number
- Single local ID name
- Local user name that matches each remote user's name

You can set mapping rules for group permissions in the same way. Users and groups are mapped independently. If there are exceptions to the default mapping, you can *exclude* certain users and groups, so that they only have special guest permissions.

- Specific mapping

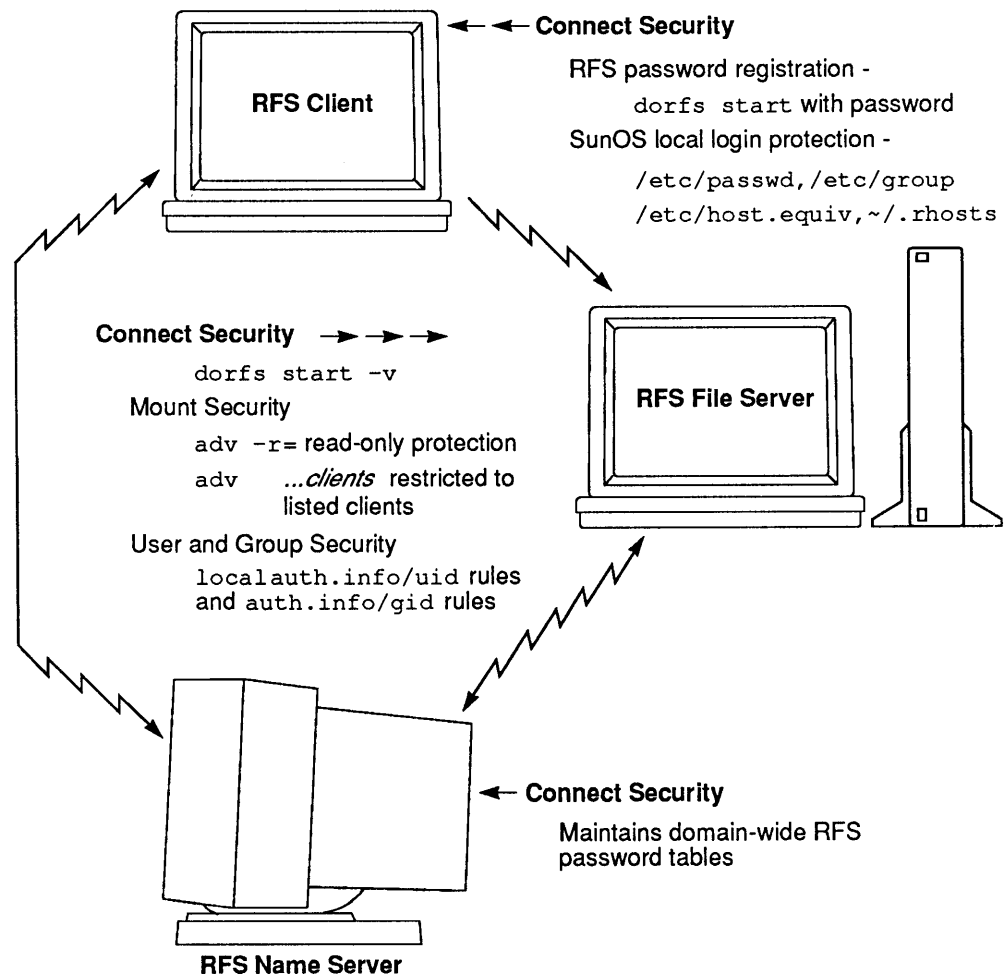
You can map any user or group from any remote machine into a specific user or group on your machine. You can do this by user name or user ID. (You cannot use remote names in global blocks.)

Using these mapping techniques and traditional SunOS file permissions, you can keep strict controls over your resources, even after they are remotely mounted.

Determining Security Measures for Your Machine

Each of the three classifications of RFS machines can implement all, some, or none of these security levels. Figure 18-3 shows the roles each type of RFS machine plays in the security scheme.

Figure 18-3 Security Levels on Machines Using RFS



Setting Up Connect Security for the RFS Domain

Connect security governs whether an RFS client is allowed to establish a connection to an RFS file server. It is achieved through RFS password verification. RFS passwords must be at least six characters long and contain at least one numeric character.

Some actions are required before RFS password verification can take place.

- The administrator of the primary RFS name server must register client machines' RFS passwords in the `domain/passwd` file.
- The administrator of an RFS file server must copy the name server's `domain/passwd` file onto the file server.
- The user on a remote client must start RFS and supply the RFS password for that machine.

Thereafter, any client that starts RFS and supplies the correct RFS password is allowed to connect to the protected RFS file server.

This subsection explains how to establish connect security on the three types of RFS machines. It contains procedures for administrators of the primary RFS name server, file server, and client machine. Note that if you administer your domain's primary RFS name server, you need to set up connect security before the client and file server administrators can do likewise.

Procedures for the Primary RFS Name Server

Connect security for the domain begins when you set up RFS password verification on the primary name server. It is completely optional. If you consult with other RFS administrators and users, you may decide that your domain needs password verification only for certain file servers and clients. In all cases, be aware of the sensitivity of resources advertised by file servers in your RFS domain, and which machines may or may not be entitled to use them.

If you determine that password verification is appropriate for your domain, begin by running the `rfadmin` command for every RFS client in the domain. `rfadmin` is completely described in the `rfadmin(8)` man page. Use the following form for each client to enable password verification. You must be superuser in order to run this command.

```
/usr/bin/rfadmin -a domain.client_name
```

`rfadmin` then prompts for the password of the client machine. Have the user type in the RFS password. This password is unrelated to the passwords in the `/etc/passwd` file or `yppasswd` maps. The user can choose the same password for RFS as their login password or pick a different one. If the client does not need to provide a password, simply press **RETURN** in response to the prompt.

`rfadmin` creates the `domain/passwd` file, `/usr/nserve/auth.info/rfs_domain/passwd`, if it does not exist, and stores the new passwords there. Afterward, when the client machine starts RFS, it prompts the user to supply this RFS password.

Here are procedures for setting up password verification.

1. Become superuser on the primary name server.
2. Type the following command.

```
# rfadmin -a rfs_domain.client_name
```

You will see the following prompts on your name server.

```
Enter password for client: rfs_password
Re-enter password for client: rfs_password
```

3. Have the user type the selected password in response. Alternatively, you can create an RFS password for the client or simply press **RETURN**. The user

can then supply his or her chosen RFS password when starting RFS on the client machine.

The Domain Security Directory

When you set up RFS security, the security-related files reside in the directory `/usr/nserve/auth.info/rfs_domain`. Depending on your domain's security scheme, these files may include the following.

- The domain-wide password verification file you set up using `rfadmin`.
- Domain-wide mapping rules files, explained shortly in the subsection, *Setting Up User and Group Security*.
- Optional subdirectories for each machine in the RFS domain, called `/usr/nserve/auth.info/rfs_domain/machine_name`. Each contains copies of the corresponding machine's local `/etc/passwd` and `/etc/group` files. The presence of these files depends on whether you implement certain types of user and group mapping.

For RFS security features to work throughout your domain, you must advertise `/usr/nserve/auth.info/rfs_domain`. You simply use the `adv` command, as shown earlier in this chapter. By convention, you should give the `/usr/nserve/auth.info/rfs_domain` directory the name of the RFS domain.

For example, suppose you want to advertise the security files for domain `rfsing`. Assuming that these files reside in `/usr/nserve/auth.info/rfsing`, you would issue the following command.

```
# adv -r -d "rfsing sec" RFSING /usr/nserve/auth.info/rfsing
```

In the command, the `-r` option advertises the resource with read-only permission. The directory `/usr/nserve/auth.info/rfsing` is advertised as the resource `RFSING`. Because no clients are listed on the command line, any RFS client that can access domain `rfsing` can mount this directory.

After explicitly advertising the domain security directory, create an entry in the `rstab` file so that the directory is automatically advertised during boot up.

Maintaining Password and Group Files

Depending on user and group mapping strategies, you may want to create a centralized location on your RFS name server for the local `/etc/passwd` and `/etc/group` files of RFS clients

Should you decide to do this, the first step is to create a subdirectory under `/usr/nserve/auth.info/domain_name` for each host in your RFS domain. Give the subdirectory the name of the host.

Then you can take either of the following actions. Have each RFS client and file server send you a copy of their local `/etc/passwd` and `/etc/group` files. Each user should do the following on their client and file server machines.

1. Send the name server a copy of the `/passwd` database.


```
# rcp /etc/passwd name_server:/usr/nserve/auth.info/domain_name/machine_name
```

where *machine_name* is the name of the client's machine.

2. Repeat the same `rcp` command, but replace `/etc/passwd` with `/etc/group`.

Alternatively, as RFS domain administrator, you can do the following.

1. Advertise the `/usr/nserve/auth.info/domain_name` directory on your name server with read-write permission.
2. Have the administrators of the remote hosts create the subdirectories for their machines in the domain security directory.
3. Have them use `cp` to add their local `/etc/passwd` and `/etc/group` files to the subdirectories they just created.

Setting Up Password Verification on the File Server

This section explains how to ensure connect security on your RFS file server by setting up password verification. If you determine that all your resources are sensitive, it is wise to set up password verification as the first level of protection.

The primary RFS name server keeps records of RFS machine passwords and user and group information for your RFS domain. These are stored on the name server in the security directory for the domain,

`/usr/nserve/auth.info/domain_name`. When the name server's administrator sets up password verification, he or she will advertise the security directory as a resource with the same name as your domain. To verify that the security directory has been advertised, simply run `nsquery` for your domain.

Once you verify that the resource has been advertised, you must mount it and copy the directory structure into `/usr/nserve/auth.info/domain_name` on your RFS file server. You can unmount the resource after copying is complete. However, if a machine in your RFS domain changes its RFS password or user and group information, the directory on your file server will be out-of-date. Therefore, you will have to re-copy the changed files yourself, or set up a script that automatically copies changed files from the primary name server.

Copying the Domain Security Directory

Here is a procedure for copying the security directory from an RFS primary server to your file server.

1. Become superuser on your RFS file server.
2. Create a security directory on your machine.

```
# mkdir /usr/nserve/auth.info/domain_name
```

where *domain_name* is the name of your domain.

3. Go to the new directory and copy the name server's security directory there.

```
# cd /usr/nserve/auth.info/domain_name
# rcp -r name_server:/usr/nserve/auth.info/domain_name .
```

Now you will have all the password and user mapping information for your domain existing locally on your file server. If you communicate with other RFS domains, you should mount the *domain_name* resource for those domains, and copy their contents as shown above.

Automatic RFS Password Verification

Once the */usr/nserve/auth.info/domain/passwd* file (hereafter called *domain/passwd*) is in place, your file server automatically verifies the password of any machine trying to mount its resources. Here is what happens.

1. A remote client tries to mount your resources.
2. If the client has an RFS password, your file server verifies it against the corresponding entry in your machine's *domain/passwd* file.
3. If the RFS passwords match, your RFS file server allows the remote machine to connect. If the RFS passwords do not match, the client cannot connect to your server.

If a client without a password tries to mount your resources, your file server grants it connect permission.

This may be all the password verification necessary for your resources. However, you may want to additionally restrict your resources only to machines with passwords in *domain/passwd* or, perhaps, only certain machines in that file. To implement this more stringent security measure, do the following.

1. Unadvertise your resources and stop RFS on your file server.

```
# dorfs stop
```

2. Start RFS as follows.

```
# dorfs start -v
```

Thereafter, only clients with RFS passwords that correspond to their entries in your *domain/passwd* can connect to your file server. Remote clients without RFS passwords or passwords that do not match the server's corresponding entries are denied access.

To restrict your resources even further, you can limit the machines allowed to connect to your server to a subset of the entries in *domain/passwd*. You must manually edit that file and delete entries for machines to which you want to deny access.

Caution: Do not delete entries from *domain/passwd* if your file server also functions as an RFS name server. You must maintain a complete list of RFS passwords for your domain.

Procedures for the RFS Client

Starting RFS password verification on a client machine is simple. The administrator of the RFS name server will ask you for your machine's RFS password or give you a password for that purpose. This password has nothing to do with your login password. You can use the same password or choose a different one.

If your machine is a brand new RFS client, make sure to install RFS as explained in the section, *Starting RFS on a Client*. If your machine already runs RFS, you will have to stop it as follows.

```
# dorfs stop
```

Now, start RFS.

```
# dorfs start
rfstart: Please enter machine password:
```

Supply the same RFS password that you provided to the administrator of the primary name server. This password will be compared to the password entered for your client on the primary name server. If the two passwords are in sync, your machine will save your password in the local password file, `/usr/nserve/loc.passwd`. You will not have to enter it again. Thereafter, you can automatically start RFS from the `/etc/rc` file, when your machine boots.

If you run `dorfs start` and your password does not match the one for your client on the current RFS name server, you will receive a warning, but `dorfs start` will NOT fail. Though RFS will run, you may have a problem if the `domain/passwd` file on the primary RFS name server is shared with other machines to use for verification. In that case, your remote mount requests will fail if the passwords do not match. For this reason, you should coordinate with the RFS domain administrator to keep your client's RFS password in sync with the primary name server. If passwords aren't important in your environment, you can simply press **RETURN** for the passwords on the client and the primary name server.

If you receive warnings that your password is out of sync, your next recourse depends on whether the primary or a secondary is the current RFS name server.

- Secondary is the current name server

If the primary went down and a secondary took over an RFS name server, the secondary may not have a domain password file or may have one that is out of date. In this case, do not try to correct your password until the primary takes over an RFS name server again.

- Primary is the current name server

Try to correct your password by re-entering it with the `rfpasswd` command. Simply type `rfpasswd` and respond with your password. If that does not work, follow these steps.

1. Have the administrator of the primary name server type the following.

```
# rfsadmin -r domain.client_name
# rfsadmin -a domain.client_name
```

where *domain.client_name* is the name of your domain, a period, then the name of your client.

2. Type the password on your client machine in response to the prompt.

```
Enter password for client_name:
```

3. Become superuser on your machine and type the following.

```
# rm /usr/nserve/loc.passwd
```

This removes the entry in */loc.passwd*, which is inconsistent with the name server.

4. Restart RFS by typing this.

```
# dorfs start
rfstart: Please enter machine password:
```

In response to the prompt, type the password you just entered on the primary name server.

You should then make sure that any RFS file server that verifies your machine's password copies the new domain password file from the primary.

Setting Up Mount Security

Once remote clients are allowed to set up a connection to an RFS file server, mount security becomes the resources' second line of defense. The administrator of the RFS file server uses options of the *adv* command to set up mount security.

If you are the administrator of an RFS file server, you can use the *clients* argument of *adv* to limit resources to individual machines, groups, or domains. Conversely, by not providing a *client* argument to *adv*, you grant mount access to the public.

The earlier section, *Using the adv Command*, explains how to set up the security options of *adv*. In addition, you can consult the *adv(8)* man page for more information.

Setting Up User and Group Security

User and group security is the final level of protection for your resources. It enables the administrator of an RFS file server to define the permissions remote users will have once they access the server's resources. If you administer a file server, you can set this up by mapping remote user IDs or user names into user IDs or names defined on your machine. Likewise, you can map remote groups into groups on your file server.

You do not have to map remote users. By default, all remote users are mapped into a *special guest ID number*, which is MAXUID plus one. MAXUID is the maximum ID number defined for the system. By default, MAXUID+1 is 65534, the “nobody” system account.

MAXUID+1 is always guaranteed not to overlap with any current or future users.

How Remote User Mapping Works

When you set up user and group mapping, you define how your RFS file server will handle requests from users and groups on remote RFS clients. This mapping impacts not only the remote users’ access to your machine’s resources, but also each remote user’s view of ownership.

For example, assume that you map user ID 101 from machine kulthum into user ID 115 on your RFS file server. User 101 on kulthum tries to create a file called “jasmine” in the directory called “perfume” on one of your advertised resources. Your file server will translate the request from 101 into a request from 115. If 115 has permission to create a file in that directory, then the file will be created.

Now suppose you run `ls -l` on your file server, on directory perfume with the newly-created file.

```
% ls -l perfume
drwxr-xr-x  2 steff          512 Apr 29  1988 ECDperf/
drwxrwxrwx  2 amina        1024 Apr 13  1988 adminperf/
-rw-r--r--  1 115          2211 Sep 16  14:30 jasmine
drwxrwxrwx  4 judy         1024 Sep 27 12:17 attperf/
-rw-r--r--  1 nat          1115 Aug 10  1989 beg.perf
```

The display shows user ID 115 as the owner of file jasmine.

On the other hand, if a user on machine kulthum runs `ls -l` on the perfume directory, your file server does *inverse mapping*. Therefore, the user on kulthum sees the file as being owned by user ID 101.

```
% ls -l perfume
drwxr-xr-x  2 steff          512 Apr 29  1988 ECDperf/
drwxrwxrwx  2 amina        1024 Apr 13  1988 adminperf/
-rw-r--r--  1 101          2211 Sep 16  14:30 jasmine
drwxrwxrwx  4 judy         1024 Sep 27 12:17 attperf/
-rw-r--r--  1 nat          1115 Aug 10  1989 beg.perf
```

Inverse mapping from the RFS file server that owns the resource provides the most consistent file system view to a remote user. It could potentially cause confusion, though.

Now, suppose that instead of just mapping 101 into 115, you also mapped user ID 102 from client kulthum into 115 on your file server. A file created by 102 would correctly create the file as owned by 115 on your machine. However, when a user on kulthum runs `ls -l` on the file, it would always show ownership by the smaller numeric value: user ID 101.

Note: This same result would occur if you gave several local user names the same numeric ID.

If users complain that files they create do not appear to belong to them, the situation described above could be the reason. This does not cause any problems with each user's ability to access the resource. However, it could break some programs that are dependent on local IDs.

Defining Mapping Rules

If you decide to set up special user and/or group ID mapping on your RFS file server, you must define mapping rules. This involves the following generic steps.

1. Obtaining the local `/etc/passwd` and `/etc/group` files from the remote clients.
2. Creating a subdirectory of `/usr/nserve/auth.info/domain_name` named for each remote client, then storing the individual client's local `passwd` and `group` files there.
3. Creating the `uid.rules` file for user mapping rules and the `gid.rules` file for group mapping rules in the directory `/usr/nserve/auth.info`.
3. Running the `idload` command to do the user and group mappings. This command reads the user and group mapping rules you create, reads your file server's `/etc/passwd` and `/etc/group` files, if needed, and maps the remote users into your file server's users' permissions. `idload` reads the user and group lists you obtained from the remote hosts, then translates those names into the appropriate numeric ID numbers.

The `idload(8)` man page contains details about the `idload` command and the `uid.rules` and `gid.rules` files.

Rules Files Syntax

The rules files `uid.rules`, and `gid.rules`, tell `idload` how to map remote users. You created these files in the `/usr/nserve/auth.info` directory when you first started RFS on the file server.

The example below shows the syntax for a `uid.rules` file. You use exactly the same syntax for the `gid.rules` file. All lines in each file are optional.

```
global
default local_id | transparent
exclude [remote_id-remote_id] | [remote_id]
map [remote_id:local]

host domain.hostname [domain.hostname...]
default local | transparent
exclude [remote_id-remote_id] | [remote_id] | [remote_name]
map [remote:local] | remote | all
```

In the syntax statement above, the italicized variables represent the following:

<i>local_name</i>	Local user name
<i>local_id</i>	Local user ID number

<i>remote_id</i>	Remote user ID number
<i>remote_name</i>	Remote user name
<i>local</i>	Local name or ID number
<i>remote</i>	Remote name or ID number

A rules file is divided into blocks of information. Each block is either a global or host block. You can designate only one global block per file, but you can set up one host block for each RFS client mapped. Note that the pipe symbol (|) denotes an “or” choice. The following bulleted items explain each line of the rules file in detail.

□ *global*

This line starts the block of global information. Each line of definitions after *global* and before the first *host* line will apply to all remote RFS clients not explicitly defined in host blocks. You can use the options *default*, *exclude*, and *map*, which are defined shortly, inside global blocks.

You cannot map or exclude names in global blocks. You must use ID numbers.

□ *host domain.hostname [domain.hostname...]*

This line starts a block of host information. Each line of definitions following this line and before the next *host* line will apply to the *domain.hostname* specified. You can use *default*, *exclude*, and *map* inside host blocks.

If you want to map more than one host from a single set of *passwd* and *group* files, you can put several hostnames on one line. In this case, *idload* will read the *passwd* and *group* files for the first host named and use the information in those files for all hosts named. A host can only be mapped once in each rules file.

□ Each of the following lines of information can appear in either a host block or a global block. A user name or ID should only be mapped once in each block. If you do map a user name or ID more than once, the first reference is in effect.

a. *default local | transparent*

You can put one “default” line in each block to indicate how to handle remote users and groups that aren’t explicitly mapped or excluded.

transparent means that for undefined users, use the same numeric ID on your machine that the user has on the remote machine. Therefore, if a request comes from remote user ID 101, your machine will give the request the permissions of local user ID 101.

local is replaced by a local user name or ID number. By default, all remote users are mapped into the permissions of the local user indicated by name or ID. If you do not specify a default line in a block, your machine assigns MAXUID+1 permission to remote users.

- b. `exclude [remote_id-remote_id] | [remote_id] | [remote_name]`

You can optionally specify “exclude” lines in a block to exempt certain users from the default mapping. For example, you can designate zero or more ranges of ID numbers using the form *remote_id-remote_id*, single *remote_names*, or single *remote_id* numbers. (You cannot specify *remote_name* in the global block.)

A user who is excluded still has access to your resources but will only have permissions of the MAXUID+1 user. You must specify all `exclude` lines before any `map` lines in a block.

- c. `map [remote:local] | remote | all`

You can use “map” lines in each block to assign local permissions to particular remote users. There are several ways to use the `map` command. For example, you can set any remote user’s permissions to any local user’s permissions by either user *ID* or *name*; separate the two with a colon (:). By entering a single *remote_id* or *remote_name*, the remote user who matches will have the permissions of the local user of the same ID or name. The example below gives the remote user `mcn` the same permissions of local user

```
map mcn
```

The literal entry `all` maps all users by user name into the permissions of users with the same name on your host.

Note that `map all` and mapping by name are not allowed in a global block. `map all` usually produces error messages. This is because multiple administrative logins have user ID 0, and `idload` will try to map each one 0 to 0. There is no harm in this.

The `idload` Command and Mapping

Once the rules files are created, the `idload` command reads your rules files and maps the users into your host. The syntax of `idload` is below.

```
idload [-n] [-g g_rules] [-u u_rules] [directory]
```

The options are defined as follows.

- `-n` This is the “no update mode” option. When it is used, `idload` will print its results in your console window without creating translation tables.
- `-g g_rules` This option lets you use a group rules file other than `/usr/nserve/auth.info/gid.rules` as input for group mapping rules.
- `-u u_rules` This option lets you use a user rules file other than `/usr/nserve/auth.info/uid.rules` as input for user mapping rules.

directory This option indicates that some directory other than `/usr/nserve/auth.info` contains the *domain/host* directories where the user and group files for each remote host reside. If it is not used, `/usr/nserve/auth.info` will be searched for the *domain/host* files.

`idload` runs automatically on both the local and remote hosts during a mount. However, you can run `idload` manually with the `-n` option. This option is useful for making sure you have done your mapping correctly. Each time you set up or change your rules files, first run `idload -n`. The results will show you the mapping that will occur when the command is run to actually load the IDs.

Choosing Mapping Strategy

This section describes some strategies you can use to map users. It describes the easiest way to deal with remote user permissions and progresses to the most complicated ways. Read through each example to decide what strategy is best for your RFS file server.

No Mapping

If you do not run `idload` to map users, all remote users have the permissions of the special guest user ID number, `MAXUID + 1`. Because no one has that user ID, guest users on your local machine only have the same file permissions granted to the public.

Mapping Remote IDs

If you map remote users using remote ID numbers and local ID numbers and names, you do not need to get any `passwd` and `group` files from remote hosts. Here are some simple examples from `uid.rules` files that only involve mapping remote ID numbers.

```
global
default transparent
exclude 0
```

Note: The `exclude 0` line is strongly recommended to prevent possible security breaches from root users on other systems.

In this example, all remote user IDs are mapped into the same user ID permissions on your host, except for `root` (ID number `0`). `root` only has special guest permissions. The mapping in this `uid.rules` file applies to all remote hosts.

```
global
default transparent
exclude 0-100
map 732:106
```

In the example above, users have the same permissions as in the previous example, with these exceptions.

- User IDs 0 through 100 have `MAXUID + 1` permissions
- Any remote user ID 732 has the same permission as local user ID 106.

```

global
default transparent
exclude 0-100
map 732:106

host rfsing.placido
default mpg
exclude 0-50

```

Here, any remote users on RFS client placido in domain rfsing will not be mapped according to the global rules applicable to other machines in the domain. Instead, users on placido have the permissions of local user mpg, except user IDs 0 through 50, who have MAXUID +1 permissions.

Mapping Remote Names

To map specific remote user names into user names local to your RFS file server, you must have access to the `/etc/passwd` and `/etc/group` files from the other machines in your RFS domain. These files reside in subdirectories of the security directory resource advertised by your RFS name server. The earlier subsection, *Setting Up Connect Security for the RFS Domain*, describes how to obtain these directories. Once you copy them, they reside in the directory `/usr/nserve/auth.info/domain_name/client_name`, where *domain_name* represents your domain name, and *client_name* represents the remote RFS client.

`idload` consults the remote client directories when it finds a request for a remote user name in a *domain.host* information block. It checks for the remote machine's `passwd` and `group` files.

The next paragraphs contain examples of `uid.rules` files that illustrate mapping remote user names. Note that mapping by name can be a very useful feature. However, if you map only by ID number or local name and avoid mapping by remote names, you also avoid the need to coordinate distributing and updating remote `passwd` and `group` files and rerunning `idload`.

`map all`

If you have the same set of user names on different machines, but the user IDs differ, you may want to use `map all` as shown below.

```

global
default transparent
exclude 0

host rfsdance.raks
exclude azhar 0 uucp
map all

```

In the above example, a remote user on RFS client raks in domain rfsdance can access your resources with the same permissions as the user name on your host that corresponds with their user name on raks. The only exceptions are anyone accessing your resources from raks with user names `azhar`, `root`, and `uucp`.

Your RFS server will grant these users MAXUID + 1 permissions.

map name:name

You can selectively map certain remote user names into local user names or user IDs on your host. Here is an example.

```
global
default transparent
exclude 0

host rfsdance.raks
default transparent
exclude 0
map amina:jg stefania shara:103
```

Here all users accessing your resources from RFS client raks are mapped into their corresponding user IDs on your machine, with the following exceptions. Remote user amina has the permissions of local user jg, remote user stefania has the permissions of local user stefania, and remote user shara has the permissions of local user ID 103.

18.7. Maintaining RFS Services

RFS maintenance tasks fall into three areas, machine administration, domain administration, and recovery. The following machine and domain administration sections describe how to maintain your RFS machine and domain. The section on recovery describes how to recover the RFS environment should a name server go down.

Administering RFS on individual machines requires little maintenance. Typically, you use the following commands, many of which have already been introduced.

- `dorfs start` starts up RFS.
- `dorfs stop` shuts down RFS.
- `idload` prints and updates remote user permissions to your resources.
- `rpasswd` changes your RFS password.
- `dname` prints and changes your machine's RFS domain name.
- `rfsadmin` lists the current name server for your domain.

Once you set up RFS, you rarely need to use the administrative commands directly. If you administer an RFS file server, set up your `rstab` and `/etc/rc` files so that your machine will automatically start RFS and advertise resources upon boot up. Refer to the section, *Setting Up an RFS File Server* for more information. On an RFS client, add RFS mount entries to the `/etc/fstab` file to mount RFS resources during booting. Other commands—`idload`, `rpasswd`, and `dname`—are used to alter the existing RFS set up.

If you are the RFS domain administrator (typically in charge of the primary name server), you additionally may use the following commands:

- `rfsadmin -a`, which adds a new host to a domain.
- `rfsadmin -r`, which removes a host from a domain.
- `rfsadmin -p`, which resets the current name server.

You issue these commands on the primary name server.

In addition, you have to maintain the `rfsmaster` file and make sure user and group information for your domain is up-to-date.

There are two commands used for monitoring RFS activities on all machine types.

- `fusage`, which prints the kilobytes of data transferred, read, and written to disk by remote hosts as well as by local users.
- `df`, which lists disk space available for a resource.

You can use the information produced by these commands to make decisions about load balancing and parameter tuning.

Maintaining RFS for RFS File Server and Client Machines

Maintaining RFS on a client machine requires little more than knowing how to start it up and stop it. Here is a quick way to familiarize yourself with the client's RFS environment.

domain name	Type <code>dname -a</code> to see the name of your host's domain and the type of network it is running on.
current name server	Type <code>rfsadmin</code> to print the hostname of the current name server.
mapping rules	Type <code>idload -n</code> to see the user mapping that is done, based on the <code>uid.rules</code> and <code>gid.rules</code> files in <code>/usr/nserve/auth.info</code> . They are optional, however, and therefore they may not exist.

Below are the RFS tasks you may need to perform on your host.

Starting Up a Client

You start up RFS activities by running `dorfs start` on your client machine. This command does the following:

- Starts up the `listener` daemon
- Mounts any RFS resources it finds in the `fstab` file.
- Registers the client with the RFS name server.

Type the following to initially start RFS services.

```
# dorfs start
```

If RFS does not start, try checking the following.

- Is the RFS name server (primary or secondary) up and running?

Try using the `ping` command to find out the name server's status.

```
% ping name_server
```

to find out the name server's status.

- Is the `listener` daemon present on your machine?

Each RFS machine on the network must have a `listener` daemon running. The `listener` handles RFS requests received over the network. Normally, each machine starts the `listener` when you run `dorfs start`. Issue the next command to see if the `listener` is running.

```
# ps -aux | grep listener
```

If the result of `grep` is only `grep listener`, you will have to restart the `listener`.

- Is your network media properly connected?

Check the underlying network hardware, such as Ethernet cabling and controller boards, for loose connections and other problems.

- Are you superuser?

You must be logged in as root.

Running `dorfs stop` to Shut Down RFS

The `dorfs stop` command stops the `listener` daemon and disconnects your machine from the RFS domain. `dorfs stop` forces remote clients to unmount all advertised resources that your machine may share (if it is an RFS file server) and unmounts any remote resources your machine may receive as a client.

Changing the RFS Password

The `rpasswd` command changes your machine's RFS password, both locally and on the primary RFS name server. Processing of the new password follows the same criteria as the SunOS `passwd` command.

Since changing passwords requires your machine to communicate with the primary name server, RFS must run on both your host and the primary RFS name server. You cannot change your RFS password if the primary is down and a secondary is the current RFS name server.

CAUTION

When you change your RFS password, machines that are authenticating your machine may not automatically receive the change. If communication fails with another machine after you change your RFS password, check that they have copied the latest version of your RFS domain's `domain/passwd` file from your primary name server.

Updating User and Group Information

The `idload` command updates remote user permissions to your resources. If your machine gains new local users and groups, you may want to change its mapping rules, if you are mapping remote names or specific user IDs.

You should regularly check to see if a machine whose users you map by name has updated its `RFS passwd` and `RFS group` files on the name server. If so, you can use `idload` to re-map that information into your machine. The `idload`

`-n` command is useful for checking the mapping that is currently active. For a detailed description of `idload`, see the “Mapping Remote Users” section or the `idload(8)` man page.

18.8. Domain Maintenance

If you are an RFS domain administrator, familiarize yourself with the basic setup of your RFS name servers and the simple maintenance tasks that you have to perform. Your RFS domain will have one primary and possibly one or more secondary name servers. The primary will keep authoritative records for the domain.

If the primary goes down, a secondary RFS name server will quickly take over. A secondary only has limited responsibilities. You cannot add or remove machines from the `domain/passwd` file on a secondary, for example. RFS name server responsibilities should be passed back to the primary as soon as it comes up.

As RFS domain administrator, you can use `unadv` to unadvertise any resource within the domain. (Only use `unadv` when an RFS name server has gone down; otherwise the domain and host advertise tables will be out of sync.)

```
unadv domain.resource
```

Specify *resource* to unadvertise one of your machine’s resources, and *domain.resource* to unadvertise any resource in an RFS name server’s domain.

Here is a summary of information you should know about your RFS domain and your name server.

Domain name	Type <code>uname -a</code> to see the name of your name server’s domain and network type.
Name servers	Look at the <code>rfmaster</code> file in <code>/usr/nserve</code> . This file will list all primary and secondary name servers in the RFS domain(s) your machine serves.
Current name server	Type <code>rfadmin</code> to print the host name of the current RFS name server. (Make sure this is your machine before you do any RFS domain administration tasks.)

The next subsections describe basic RFS domain maintenance tasks that you perform on a name server.

Adding New Machines to `domain/passwd`

You will find instructions for adding new clients in the previous section, *Setting Up RFS Security*.

Removing a Machine from `domain/passwd`

To permanently remove a machine from an RFS domain, you must become superuser on the primary name server and type this command.

```
# rfadmin -r rfs_domain.machine_name
```

If *machine_name* is not listed in the `rfmaster` file as a primary or secondary name server, `rfadmin` will remove the entry for it from the `domain/passwd`

file. Otherwise, you will receive an error message, and the host will not be removed.

Change Current RFS Name Server

If your host is the current RFS name server, you can pass that responsibility to another name server, as follows.

```
rfadmin -p
```

The `rfadmin -p` command passed RFS name server information to the next name server listed in the RFS domain's `rfmaster` file. That machine will then act as the current RFS name server. The major use of the `-p` option is to return name server responsibility back to the primary after it has gone down and come back up. As the administrator of the primary name server, you may also want to do

Because a secondary is only intended to take over RFS name service temporarily, you should use the `rfadmin -p` command to pass name server responsibility back to the primary as soon as possible. It does not happen automatically. You cannot perform most RFS domain maintenance, such as adding new hosts or changing passwords, while the secondary is acting RFS name server.

Reassigning RFS Name Service

If you want to reassign RFS name server responsibilities on a more permanent basis, you have to change the `rfmaster` file in `/usr/nserve` as follows.

- Stop RFS on all primary and secondary RFS name servers by running `dorfs stop`.
- Change the `rfmaster` file on the primary RFS name server.
- Start the primary name server again by running `dorfs start`.
- Restart any secondary name servers by running `dorfs start`.

Once you change the information on the name servers, each RFS client will pick up the change the next time it starts RFS.

Adding New RFS Domains

If there are other RFS domains in your network, you may want machines in your RFS domain to communicate with them. To do this, get the names and IP addresses of the RFS name servers for each domain and add them to the `rfmaster` file in `/usr/nserve`.

Updating User and Group Files

If machines in your RFS domain map remote users by name, the primary name server can provide a central point for gathering machines' RFS password and group information. Users or administrators on each RFS client must provide you with this information. When you receive updated RFS `passwd` and `group` files, install them in the directory for that machine. The RFS `passwd` and `group` files will go into the proper `/usr/nserve/auth.info/domain/domain/host` directory.

18.9. Recovery in the RFS Environment

As a member of an RFS domain, your computer relies on other machines in the domain for resource sharing. RFS is designed to recover quickly when communication is cut between machines, either because an RFS name server or file server went down or withdrew its services. The following sections describe RFS events that can occur and the recovery mechanisms designed to handle them.

Primary Goes Down

The primary RFS name server maintains all essential RFS domain records. This machine regularly distributes the most critical of these records to the secondary name servers. (These records do not include files and directories under `/usr/nserve/auth.info`.)

If the primary goes down, RFS name server responsibilities are passed to the first secondary name server listed in the `rfmaster` file. The secondary is only intended to take over temporarily, because it has limited name service capabilities. This is done to maintain the definitive RFS domain records on the primary. Changing the RFS name server does not affect any currently mounted resources.

While a secondary is acting RFS name server, you cannot perform the following functions.

- Maintaining host lists
You cannot add or delete machines from RFS domain lists while a secondary is acting RFS name server.
- Changing host passwords
Neither the secondary nor another machine can change RFS passwords while a secondary is acting RFS name server.

The secondary will maintain lists of advertised resources for the RFS domain and continue basic name server functions, so that RFS activities can continue. In most cases, members of the RFS domain should not be aware that the primary name server is down. When the primary comes back up, the administrator of the secondary should pass name server responsibilities back to the primary by using the `rfadmin -p` command.

Note that when a primary name server crashes without properly shutting down RFS and passing name server responsibilities to a secondary in an orderly fashion, the advertise table on the secondary may contain some errors. The advertise table may list resources from the primary as available, and recently advertised resources from other RFS file servers may not be listed. You can fix the RFS domain advertise table using `unadv` and `adv -m` commands from the RFS name server.

Primary and Secondaries Go Down

If all primary and secondary name servers go down at once, all information on advertised resources will be lost. Active mounts and links, however are not disturbed. However, when the primary comes back up, each RFS file server will still think its resources are advertised, but the primary will have no record of them.

As soon as the primary is running, each RFS file server can make sure its advertised resources are in sync with those listed on the primary by doing either of the following.

- Restarting RFS

Run `dorfs stop`, followed by `dorfs start`.

- Re-advertising resources on the RFS file server

This is a less drastic way to update the advertise tables on the primary name server. Re-advertise each resource using the `adv -m` command from the RFS file server where the resource resides. This command will get the primary and file server advertise tables back in sync.

RFS File Server Goes Down

When an RFS file server goes down, clients that have mounted its resources lose contact with those resources. The remote file sharing daemon process `rfudaemon` runs an administrative shell script `rfuadmin` to handle user-level functions that are required when a link goes down.

`rfudaemon`

The `rfudaemon` process is run automatically when you run `dorfs start`, and continues to run until you run `dorfs stop`. The `rfudaemon` process waits for one of the following events to occur, then passes that information to the `rfuadmin` administrative shell script.

<code>disconnect</code>	When a link is cut to a remote resource, <code>rfudaemon</code> sends a <code>disconnect</code> message and the resource name to <code>rfuadmin</code> .
<code>fumount</code>	When a resource is unmounted by the RFS file server, the <code>rfudaemon</code> sends a <code>fumount</code> message and the resource name to <code>rfuadmin</code> .
<code>fuwarn</code>	When a RFS file server sends a message that it is about to unmount a resource, the <code>rfudaemon</code> sends a <code>fuwarn</code> message to <code>rfuadmin</code> , giving the resource name, and the number of seconds before the resource will be unmounted.

`rfuadmin`

When links to resources are disconnected, the response to the `disconnect` is handled at the user level by `rfuadmin`. By editing this shell script, you can tailor the response your system makes when a resource is cut. The `rfudaemon` process starts `rfuadmin` with one of the following arguments.

`disconnect resource`

When `rfuadmin` is started by `rfudaemon` with these arguments, `rfuadmin` sends this message to all terminals using `wall`.

`resource` has been disconnected from the system.

Then it executes `fuser` to kill all processes using the resource, unmounts the resource to notify the kernel, and runs `mount -o bg` to try to remount the resource. The assumption is that the link was either broken by

mistake or that as soon as the RFS file server makes the resource available again, the client will want to mount it.

`fumount resource` When `rfuadmin` is started by `rfudaemon` with these arguments, the processing is similar to a disconnect.

`fuwarn resource seconds`

When `rfuadmin` is started by `rfudaemon` with these arguments, `rfuadmin` sends this message to all terminals.

`resource` is being removed from the system in # seconds.

There are many reasons you may want to change the `rfuadmin` shell script. If access to a resource is lost, you may want to respond by trying to mount another resource. You may want to send different messages when a resource is lost.

When a resource is disconnected, `rfuadmin` tries to remount the resource using `mount -obg`, as described in the `mount(8)` man page. This command retries the remount until it succeeds. To change this behavior, edit `/etc/rfuadmin` so it no longer runs `mount -obg` or retries only a limited number of times.

Administering C2 Security

19.1. Introduction

Computer security is a growing concern. A major reason for this is that it is possible for knowledgeable but unauthorized individuals to gain access to computers and manipulate the information contained therein. The manipulation may result in the deletion of the information or in its being stolen. Information can be stolen without being physically removed from a computer file.

Password safeguarding and auditing of security-related events are two of the most important extensions that the C2 specification offers to the SunOS operating system. This chapter examines the steps needed to set up and administer C2 security.

This chapter is divided into the following sections:

- **What is C2 Security?**— The place of C2 Security among the seven NCSC security criteria
- **Setting up C2 Security**— What is involved in setting up C2 security
- **Password and Group Security**— What happens with password and group security when C2 is in place
- **Definition of Auditing Terms**— some of the most important terms used later in the chapter are presented and defined
- **Running C2conv**— A summary of what happens when you run C2conv, and a list of items you should prepare before running it
- **Sample run of C2conv**— A close examination of a sample run of C2conv, and an exhaustive explanation of the issues involved
- **After running C2conv**— A summary of the tasks that you have to perform after having run C2conv
- **Changing the Audit State**— How to change the audit state, either for the system or for a particular user, once C2 security is in place
- **Looking at the Audit Trail**— How to examine the audit trail and how to extract the data needed from it
- **When Audit File systems Are Full**— What to do when you run out of space and the system hangs

- **Running C2unconv**— What does C2unconv do and not do

Additionally, Appendix D, *The Orange Book*, compares C2 security in SunOS 4.1 to the requirements listed in “The Orange Book,” and Appendix E, *Format of Audit Records* describes the format of different types of audit records.

19.2. What is C2 security?

The *National Computer Security Center* (NCSC) has defined a set of standardized criteria for evaluating computer security. According to the NCSC, the trusted computer system evaluation criteria classify systems into four broad hierarchical divisions of enhanced security protection. They provide a basis for the evaluation of effectiveness of security controls built into automatic data processing system products. The criteria were developed with three objectives in mind:¹

- Provide users with a yardstick with which to assess the degree of trust that can be placed in computer systems for the secure processing of classified or other sensitive information
- Provide guidance to manufacturers as to what to build into their new, widely-available trusted commercial products in order to satisfy trust requirements for sensitive applications and
- Provide a basis for specifying security requirements in acquisition specifications

Levels of Security

The following are the seven NCSC security criteria:

- D Minimal protection. The system uses no special security features at all. (An example would be a personal computer in an unlocked room.)
- C1 Discretionary access control. The system requires a login/password procedure, and provides access permissions based on user, groups, and others. (For example, an ordinary UNIX system.)
- C2 Auditing and authentication. Security-related events are audited, the login/password procedure provides certain authentication and encrypted passwords are stored in a place inaccessible to an unprivileged user. (For example, SunOS Release 4.1 with the Security option installed.)
- B1 Mandatory access control and labeled output. File access is based on labels indicating security clearance, and all output is labeled at the security level. (For example, SunOS MLS Release 1.0.)
- B2 Configuration control, trusted facility management, no covert channels. System configuration must be fully documented and controlled. Administrative, security, and operator functions are separate. There can be no security holes.

¹ Refer to the *Department of Defense Trusted Computer System Evaluation Criteria*, CSC-STD-001-83, and to Appendix D *The Orange Book*.

B3 Access control lists, internal structure, full documentation. File access is based not only on labels, but also upon lists of users with and without access to an object. The internal structure of the system must be fully documented.

A1 Formal proofs are required. Currently there are no systems available at this level.

The SunOS operating system provides a security option to match the C2 classification level defined by the NCSC. The C2 category adds password hiding and security auditing to the standard UNIX system. To incorporate C2 specifications, SunOS Release 4.1 provides improved password security, the capability to audit nearly all system events, and password requirement for single-user booting. Note that the SunOS C2 security features differ slightly from what would be required for an NCSC-evaluated C2 system; it has not been, and will not be, submitted for NCSC evaluation.

19.3. Setting Up C2 Security

There are two distinct activities required for establishing C2 security in your system or network:

- Extracting from tape all the programs needed
- Converting to C2

C2 Required Programs

C2 requires certain programs in order to run. These programs are installed when you chose the `Security` option during `SunInstall`. These programs (which reside in `/usr/etc`) and their functionalities are defined in the following chart:

Table 19-1 *List of Required Programs*

<i>Program</i>	<i>Functionality</i>
C2conv	Shell script: Converts standard passwd and group files into the secure format, and sets up auditing information.
C2unconv	Shell script: Converts secure passwd and group files back to the standard format.
c2convert	Utility: supports C2conv.
audit	User interface to the audit daemon: Allows specification of where audit trails are written, changing user audit state, etc.
praudit	Takes audit trail as input and formats it into human-readable output.
rpc.pwdauthd	Authentication daemon: Provides user and password authentication without exposing encrypted passwords. It is activated in /etc/rc.local if the password file has been split by C2conv.
*audit_warn	Shell script: warns the administrator when an audit file system becomes nearly full and when other problems occur.
*auditd	Audit daemon: responsible for writing audit-trail files.

(Programs prefixed with * are always installed, whether or not the Security option is chosen at installation time.) The following header files also are installed in /usr/include:

Table 19-2 *List of Header Files*

<i>File</i>	<i>Functionality</i>
<code>grpadj.h</code>	Describes format of <code>group.adjunct</code> file.
<code>pwdadj.h</code>	Describes format of <code>passwd.adjunct</code> file.
<code>aevents.h</code>	Describes events that should be audited.
<code>sys/audit.h</code>	Describes the format of audit records.
<code>sys/label.h</code>	Describes the format of security labels.

Loading the C2 programs

To load these programs and files, select the `Security` option when installing SunOS, if you are using the manual `SunInstall` program; otherwise, if you use `Quick Install`, select one of the various configurations that provide the `Security` option. To verify which configuration is appropriate, please consult the *Installing SunOS 4.1* or *Quick Install* manuals.

Kernel Requirements

Note: If you plan to use `secure NFS`, and your machine does not have a DES chip, your kernel also needs the `CRYPT` option.

The only special kernel requirement for using C2 security is that you must configure `/vmunix` with the option `SYSAUDIT`. If you are using the `GENERIC` or the `GENERIC_SMALL` kernel, this is what you get. If you have reconfigured your kernel, and you have not included this option, you must: update the configuration file to include the options `SYSAUDIT`, `UFS` and `QUOTA`, (the first requires the other two), run `config` to generate a new kernel makefile, run `make` to produce a new kernel containing security features, install the new kernel, and reboot. (For a full explanation on kernel reconfiguration, consult the chapter *Reconfiguring the System Kernel*.)

NIS Domains

If you are running NIS in your network, you can install C2 in just one machine, or in machines that are a subset of an NIS domain. However, for the C2 security features to work properly and in compliance with the NCSC requirements, you have to convert the entire NIS domain to C2.

Once you choose the `Security` option from `SunInstall`, or a configuration that includes `Security` from `Quick Install`, the appropriate NIS daemons are installed automatically. You have to take no further steps to run NIS, and the `/etc/rc.local` file is set to automatically start the `ypbind` daemon with the `-s` option for a slight increase in security.

Converting to C2

Once you have installed your operating system with the `Security` option, you have to run the `C2conv` program. This program sets up password and group security, and audit trailing. It asks you to make choices and provide information; a later section describes `C2conv` in detail.

19.4. Password and Group Security

In a regular UNIX environment, the `/etc/passwd` and `/etc/group` files, although relatively secure from unauthorized modification, are readable by all users; this exposes the encrypted passwords to decryption. The `C2conv` program converts the `/etc/passwd` and `/etc/group` files from the standard format into the new secure format. Copies of the old password and group files are left in the files `/etc/passwd.bak` and `/etc/group.bak`, respectively.

The file `/etc/passwd` does not contain now the encrypted users' passwords; instead, it contains the entry `##name` in the password field, where `name` is the name of the user as it appears in as the first field. The file `/etc/group` contains the entry `#$name` in the password field, where `name` is the name of the group as it appears in the first field.

The files `/etc/security/passwd.adjunct` and `/etc/security/group.adjunct` are created. They now contain the encrypted passwords, as well as auditing flags, but they are not readable by all users. Please refer to `passwd.adjunct(5)` and `group.adjunct(5)` in the *SunOS Reference Manual* for more information.

Note: For added security, you should remove the `.bak` files after verifying that C2 security works correctly, since they contain encrypted passwords for everyone to see.

19.5. Definition of Auditing Terms

Before you run `C2conv`, you should know some of the concepts related to auditing. This will make it easier for you to understand how `C2conv` works, what answers to give to its prompts, and, later, how to administer the audit trail.

Auditing does not prevent security breaches, but it provides a trail which could be useful in determining how the system was breached and who was responsible. Additionally, it can provide a wide assortment of auditing information about nearly all activities on the system.

Auditing in a C2 secure environment means that certain user actions, as defined by the *system audit value* and modified by the *user audit value* (see below), are monitored and a record of them is logged in one of the selected audit file systems. The combination of *system audit value* and *user audit value* define the *audit state*. You can change the latter at any time, either permanently or temporarily. The system audit values are defined by a series of flags in the file `audit_control` in the directory `/etc/security/audit`. The user audit values are defined in the file `/etc/security/passwd.adjunct`. These flags are set initially through `C2conv`.

The following are definitions of the most important terms used later in this chapter:

Auditing	The purpose of auditing is to gather information about: <ul style="list-style-type: none"> □ Who is performing what operations □ What operations are occurring with unusual frequency □ Who is performing abnormal operations.
System Audit Value	This is the set of audit information to be gathered for all users at login time. It is not actually kept anywhere in the system, but is used as the first step in constructing the process audit state (see below).
User Audit Value	This is the set of audit information to be gathered for a particular user ID. The state specified for a user overrides the system audit value. As examples of why the user audit value is necessary, consider these situations: <ul style="list-style-type: none"> □ The user <code>joe</code> may have been behaving oddly. It would be a good idea to monitor all his activities, but a bad idea to collect information on everyone.

- The user `fred`, on the other hand, is very reliable. Although the general audit level is high, it makes sense to reduce the data collected for `fred`.
- There is a known leak in company security. By auditing one user a day, it may be possible to discover who has illicitly obtained the superuser password.

Process Audit State Every process has an associated *audit state* that determines which events are to be audited for that process. The audit state is set at login time. It is the result of a combination of the *system audit value* and the *user audit value* according to the following rules:

- If the system audit state defines auditing for an event and the user audit state has nothing to say, the event will be audited.
- If the system audit state defines auditing for an event and the user audit state says to ignore this event, the event will not be audited.
- If the user audit state defines auditing for an event, it will be audited, regardless of the system audit state.

Since a process inherits its audit state from its parent process, all children have the same audit state as the login shell. The process audit state can be manipulated with the `audit` command. (See below, *Changing the Audit State*, `audit(8)` man page for complete details.)

Audit State Change An *audit state change* is the process of changing the audit state for some (possibly empty) set of processes. Audit state changes are useful for spot checks on individual users or specific activities. There are two kinds of state changes: permanent and immediate.

Permanent Change A *permanent change* is made by changing the administrative database. New login sessions are affected, but existing processes up to the time of a new login are not. The `login` program sets the process audit state for that process. Thus, one way to ensure that all users are running with the current audit state is to reboot the machine.

Immediate Change An *immediate* change does not modify the administrative database. It affects existing processes, but it does not affect new login sessions. You could think of this as a temporary state change. This feature is especially useful in the case of an illicit activity underway, during which the auditor would like to begin documenting the activities of a suspicious user.

Event Class An *event class* defines a set of occurrences to be audited. The classes defined to date are listed in the following table:

Table 19-3 List

<i>short name</i>	<i>long name</i>	<i>short description</i>
dr	data_read	Read of data, open for reading, etc.
dw	data_write	Write or modification of data
dc	data_create	Creation or deletion of any object
da	data_access_change	Change in object access (modes, owner)
lo	login_logout	Login, logout, creation by at
ad	administrative	Normal administrative operation
p0	minor_privilege	Privileged operation
p1	major_privilege	Unusual privileged operation

Audit Flag An *audit flag* describes a particular audit class in an audit state definition. An audit flag is an indication of what to do with an event. The format is,

```
<option><class>
```

where *option* is either +, -, or not present, and *class* is any audit class. A plus means to audit successful events. A minus means to audit failed events. The absence of either means to audit both successful and failed events.

Audit Value Definition An *audit value definition* is a comma-separated list of audit flags. Here is a sample definition:

```
+dr, -dw, lo, p0, p1
```

This means: audit successful data reads, failed data writes, all new logins, and both kinds of privileged operations.

Audit File system An *audit file system* is any file system to which audit data is written. In the Sun environment, users rarely log in and work with only a single machine. Users usually mount file systems over NFS, or use `rcp` and `rlogin`. In the workstation environment, you may want to place your audit files on a remote system mounted over NFS. This is commonly done in cases where certain central machines have greater disk resources. Typically, these are separate file systems, set aside for the exclusive use of audit trails (that is, dedicated disk partitions.) These are mounted in the directory `/etc/security/audit` of each of the client machines.

19.6. Running C2conv

The `C2conv` program uses the following files:

```
/usr/etc/C2conv
/usr/lib/c2convert
```

`C2conv` is the primary program you use to convert the system to C2 security.

A Summary of C2conv Actions

WARNING: The .bak files contain encrypted passwords. For added security, remove them after verifying that c2 security works properly.

C2conv splits the /etc/passwd and /etc/group files and sets up security auditing. You must execute it as root and while the machine is in single-user mode. Executing C2conv (which in turn executes c2convert) performs the following functions:

- Copies /etc/passwd and /etc/group files to /etc/passwd.bak and /etc/group.bak respectively.
- Places encrypted passwords from /etc/passwd in /etc/security/passwd.adjunct and from /etc/group into /etc/security/group.adjunct.
- Sets up directories for audit trail files (both local and remote), creates and mounts audit trail file systems as necessary.
- Constructs a list of events to be audited.
- Sets the minimum free space warning marker for the audit file systems (minfree variable).
- Determines whom to notify when minfree is reached (that is, when one of the file systems is getting full).
- Sets the password for user audit.
- For servers of diskless clients C2conv performs the following:
 - Compiles a list of clients to be audited and specifies their root locations.
 - Modifies each client's /etc/fstab file to mount the proper audit file system.

The following required directories are automatically generated by the C2conv shell script in the machine on which it is run:

<i>Pathname</i>	<i>Owner</i>	<i>Group</i>	<i>Mode</i>
/etc/security	root	wheel	0711
/etc/security/audit	audit	audit	0700
/etc/security/audit/ <i>server</i> /files	audit	audit	0700

In the third line, *server* should be the same as `hostname` if the audit trail is kept locally (on that machine). If all audit trails are kept on remotely-mounted file systems, no *server* will be the same as `hostname`. Instead, *server* should be the `hostname` of a remote machine.

Before Running C2conv

When you run C2conv, it prompts you for a variety of answers. The most important are:

- The path of clients' root directories
- The path of clients' executables
- The name of the audit device (disk partition) if the local machine is an audit file server, or, if not, the name of the remote audit file server, as well as the path of its audit file system(s)

- The audit files you want to establish for the system audit state

You will be well-advised to have these answers prepared in advance. If you are not familiar with the concepts, the following sample run of `C2conv` should give you a good idea of the issues.

If the local machine is an audit file server, you must remove from `/etc/fstab` any entries for local audit devices prior to running `C2conv`.

19.7. Sample Run of `C2conv`

This section closely examines a sample run of `C2conv`. It is assumed that this is done in a machine named `leonardo`, and this happens immediately after booting it in single-user mode. Each section of the run, presented in a gray box, is followed by an explanation of the issues involved.

```
# /usr/etc/C2conv
```

```
You are about to run the C2conv program. Please read the c2 security
chapter in the System and Network Administration Manual if you have not
done so yet. This program generates a shell script before it actually
affects any files. You may cancel the installation by entering Control-C
at any prompt. At the end of the procedure, you will receive a final
confirmation before applying the changes. You may then abort the
procedure and examine the generated files.
```

```
Is the system in single-user mode? [y|n]: y
```

For `C2conv` to run properly, the machine (in this case `leonardo`) must be in single user mode. If you answer `n` to the above prompt, the program will terminate, so that you can bring the machine down to single-user mode.

It is recommended that you run `C2conv` from either a file server or a standalone machine. In the case of file servers and diskless clients, both are converted to `C2` by running `C2conv` on the server. This modifies the server's `passwd` and `group` files as well as the relevant files for the clients on the server's disk. For these functions to be performed properly, you must follow this procedure:

- 1) Halt the server and its client(s)
- 2) Boot the server in single user mode
- 3) Run `C2conv` on the server
- 4) Reboot the server and its client(s) in multi-user mode

For information on halting and booting, see the chapter on *Booting Up and Shutting Down Your System*.

C2conv on a client: The best way to convert a client to `C2` is by running `C2conv` on the server. Although you can run `C2conv` directly on the client, the `audit_warn` program will not be updated, since `/usr` is mounted read-only. This should not cause a real problem. There may be a problem in doing this, however, because the permissions and ownerships of the audit directories will not be set properly. The server to which the audit files are written needs to be ready for the data.

```
Do you want audit file systems mounted using Secure NFS [y|n]: y
```

By answering **y** to the above question, you ensure that when the machine on which you are running `C2conv` (or any of its clients) mounts the audit file systems, the NFS mounting will be done using the `secure` option. For more information about NFS mounting and secure NFS, consult the chapter *The Sun Network File System Service*. Also, as mentioned above in subsection *Kernel Requirements*, the configuration files of all the machines affected should have the following line uncommented before recompiling the kernel, if the machines do not have a DES chip installed:

```
options    CRYPT    #software encryption (if no DES chip)
```

For more information on reconfiguring the kernel, see the chapter *Reconfiguring the System Kernel*.

```
Is leonardo a server for diskless clients? [y|n]: y
```

`C2conv` needs to know whether to modify the clients' files or not. Answer **y** only if the machine is a file server for diskless clients. Remember that a machine that mounts home directories from a server, but has a disk with `'/'` and `'/usr'` file systems, is a standalone, not a diskless client.

```
Enter path of clients' root (e.g. '/export/root'): /export/root
```

This prompt asks you for the path to the server's directory that contains the clients' roots, not for the path to the roots themselves. In other words, if the file server `europa` serves clients `france` and `spain`, and the clients mount their roots from `europa's /export/root/france` and `/export/root/spain` respectively, answer `/export/root`, as in the example above.

```
Enter path of additional architecture's executables
(e.g. '/export/exec/sun3') or 'done': /export/exec/sun3
```

This is the path from which clients will mount their `/usr` file systems. In a homogeneous environment the executables' path should be fairly simple: If the server and the clients run SunOS 4.1, and they are all Sun-3s, the above example would suffice. If, on the other hand, the server is a Sun-4 running SunOS 4.1 and the clients are Sun-3s and Sun-4s running both SunOS 4.0.3 and 4.1, you would have to specify all the possibilities, that is, `/export/exec/sun3.sunos.4.0.3`, `/export/exec/sun3.sunos.4.1`, and so on. Note that in this particular aspect there is no difference between `sun3` and `sun3x`, or between `sun4` and `sun4c`.

Once you have finished entering all possible paths, enter the word **done** and the menu will move to the next item.

```
Enter path of additional architecture's executables
(e.g. '/export/execs/sun3') or 'done': done
```

```
Is leonardo an audit file server? [y|n]: y
```

If none of the audit file systems reside in leonardo's disk(s), answer **n**. Otherwise, answer **y**. You should set aside at least two audit file systems for the network, so that when one fills, the system can switch over to the other and you have enough time to take appropriate action (as specified below.) The audit file systems *do not* have to reside in the same machine, that is, you may designate more than one machine as an audit file system server. This is, in fact, recommended to minimize problems if the audit file server goes down or does not respond for some reason.

```
Enter audit device (e.g. 'xyld'), or 'done': xyld
```

This prompt appears only if you answered **y** to the last question, that is, if this is an audit file server and at least one of its file systems has been set aside for recording audit data on it. Enter the name of the device (i.e. disk partition) in which the audit file system in question will reside.

If there is more than one audit file system in this machine, continue entering the name of the devices until you have exhausted them. Then enter the word **done**. C2conv modifies the server's `/etc/fstab` file, so that the device is mounted properly on the mount point `/etc/security/audit/server` (where *server* in this case is the name of the local machine). You do not have to prepare the mount point in advance: C2conv prepares it for you, with the correct ownership and permissions.

If there is more than one local audit device, the second will be mounted on `/etc/security/audit/server.1`, the third on `/etc/security/audit/server.2`, etc.

It is important to set aside enough space for audit trails, as they can expand and fill up the file systems on which they reside more quickly than expected.

If you want to do minimal logging of audit messages, you should set aside two disk partitions of at least 20 megabytes each for an 8-machine network. Two file systems are better than one, especially if they are on separate machines. Available audit space should be checked frequently at first, to determine if space needs to be expanded.

If you want to do heavy auditing, you'll need lots of disk space, and will have to dump audit trails to tape often. Unfortunately, there is no formula for determining in advance what amount of space you will need for your auditing file systems. This depends on the number of users, number of machines, level of auditing, and level of usage.

```
Enter name of remote audit file server, or 'done': willow
```

If leonardo is not an audit file server, or if it is but there is at least one other

audit file server, enter the name(s) of the other machine(s), and the word **done** when finished.

If there are no other audit file servers other than `leonardo`, enter the word **done**.

```
Enter remote audit file system on willow
(e.g. '/etc/security/audit/willow'), or 'done': /etc/security/audit/willow
```

This prompt appears only if you answered positively to the last question. `C2conv` prompts you for the name of the file system on the other audit file server (in this case `willow`). This is equivalent to the prompt for the local disk partition that you were asked above. In both cases, `C2conv` modifies the file `/etc/fstab` of servers and clients. The local device will be locally mounted on the audit file server's mount point `/etc/security/audit/server`, while the clients will do an NFS mount of that directory from the server. In turn, `leonardo` will also NFS mount `/etc/security/audit/willow`, since, from the point of view of auditing, `leonardo` is one of `willow`'s clients.

All the NFS mounts will be done with the "secure" option if you chose this option at the beginning.

```
Enter name of remote audit file server, or 'done':
```

Once you finish entering the name(s) of the audit file system(s) on `willow`, you have the opportunity to declare if there are other audit file servers in the network. If there are, you will be asked again for the name(s) of the audit file system(s) on them. If there aren't, enter **done**.

```
Specify other audit directories or 'done': /usr/tmp/security
Specify other audit directories or 'done': done
```

Notice: Clients may require mounts for these directories.

WARNING: It is very important that audit trails not be collected on the root file system, as filling this system will have unpleasant side effects.

You may also decide to have auditing written to a file system that is not completely dedicated to auditing. This is not recommended.

The Notice message is to remind you that clients of `leonardo` will not have access to this directory unless it, like all other audit file systems, is included in some exported file system. That is, the above answer (`/usr/tmp/security`) will be effective only if `willow` is exporting its `/usr` file system to the clients in question.

The above instructions assume that if you indicate that a given machine (e.g. `willow`) is a remote audit file server, the `C2conv` script has been run on that machine, or will be run on it shortly. If you have no plans to run C2 security on that machine and you are only taking advantage of its disk capacity (for testing purposes, for example) then you will have to do the following manually:

- create the appropriate `/etc/security/audit/server/files` directory on the remote audit file server

- change its owner to user audit and its mode to 700.

You are about to be asked to set the audit flags.
Do you need a summary? [y|n]: **y**

The audit flags you are about to choose are the ones that determine what event classes will constitute the *audit state* definition for all users in the system (before being modified individually for a given user, if at all). A set of flags is provided for you as a default. Depending on your needs, you may choose the default or not. The following appears only if you respond **y** to the last prompt:

The audit flags specify which event classes are to be audited. Each flag identifies a single audit class. To specify more than one flag, enter them as a comma-separated list. No spaces are allowed in your answer.

The following table lists the audit classes:

flag name	short description
dr	Read of data, open for reading, etc.
dw	Write or modification of data
dc	Creation or deletion of any object
da	Change in object access (modes, owner)
lo	Login, logout, creation by at(1)
ad	Normal administrative operation
p0	Privileged operation
p1	Unusual privileged operation

The following prefixes may be used as options:

- audit for failure only
+ audit for success only
(no prefix) audit for both successes and failures

The default audit flags are 'ad,lo,p0,p1'

The following table gives a few examples of what programs or system- or library-calls each of the flags audits. A more complete list can be obtained from Appendix E, *Format of Audit Records*.

<i>flag</i>	<i>examples</i>		
dr	stat(2)	statfs(2)	access(2)
dw	ftruncate(2)	kill(2)	utimes(2)
dc	link(2)	mkdir(2)	rmdir(2)
da	chmod(2)	chown(2)	fchmod(2)
lo	login(1)	rex(8)	rexc(8)
ad	su(1)	passwd(1)	clri(8)
p0	chroot(2)	quota(1)	quotaon(8)
p1	reboot(2)	setdomainname(2)	sethostname(2)

In addition, the special flag `all` (not listed in the above table) indicates that all events should be audited; `-all` indicates that all failed attempts are to be audited, and `+all` indicates that all successful attempts are to be audited. You should *not* use the `all` flag unless you have extraordinary reasons for it. It generates copious amounts of information that can fill up your audit file systems very fast.

The prefixes `^`, `^-`, and `^+` turn off flags specified earlier in the string (`^-` and `^+` for failed and successful attempts, `^` for both). They are typically used in `/etc/security/passwd.adjunct` to reset flags (see below, *Changing the User Audit State*).

```
ok to use audit flags 'ad,lo,p0,p1'? [y|n]: n
```

These are the default flags. If you answer `y` to this question, the audit state will specify that all data access changes, all new logins, and all privileged operations, whether regular or unusual, will be audited.

If you answer `n` you will be prompted for the flags you want to define the audit state:

```
Enter audit flags (e.g. '-ad,p0,+lo'): +lo,p0,-p1
```

This combination of flags means: audit all successful logins, all regular privileged operations, and all unsuccessful unusual privileged operations.

The combination of flags you choose, whether the default or your own, determines the initial audit state definition for all users of the system. This can be modified later for each user in their `passwd.adjunct` entry (see below, *Changing the User Audit State*). You should be guided by two criteria in your choice of initial audit value:

- The level of trust of the users on the system.
- The amount of space you have for audit file systems; the more events you audit, the more space you will need.

```
ok to use soft disk space limit of 20%? [y|n]: n
```

Setting the soft disk space limit to 20 percent means that when the current audit

file system gets 80 percent full, a mail message is sent to the designated audit administrator, and the audit trail is switched to the next audit file system, if any. This should give you enough time to make a backup tape of the audit trail, delete it, and start over again. If you find the 80 percent level too early or too late, you can change it.

The sequence of events is the following, assuming you designate two audit file systems, A and B (where A stands for `/etc/security/audit/server/files` and B stands for `/etc/security/audit/server.1/files`). When auditing starts, a file is created in A, designated `date.not_terminated.server` (where `date` is made up of year, month, day, hour, minutes, seconds). When the soft disk space limit is reached, the name of the file is changed to `date.date.server`, and auditing is switched to B, where another `not_terminated` file is created. A message is sent to you indicating that `minfree` has been reached in A. If you then free space in A, the same process will be repeated when B reaches `minfree`. Otherwise, a new `not_terminated` file is created in A, using the space left over `minfree`. When all the space in file system A is full, auditing switches to B again, until B is also full. When both file systems are full, all activities in all machines cease until space is created for auditing. The only activity that you can do is log in as user `audit` because no auditing is done on that user; you can then proceed to dump the contents of the file systems to tape and delete the files, at which time activity will resume. It is essential for `audit` to have write permission on the files and directories in question.

```
Enter soft limit percentage (e.g. '10'): 10
```

This prompt appears only if you answer negatively to the last question.

Depending on the amount of auditing you have set for your system and the number of designated file systems, 20 percent may be too high. On the other hand, if you have only one designated file system, or if you do a high level of auditing, even 20 percent may be too low.

```
ok to notify 'root@leonardo' when administration is required? [y|n]: n
```

If you answer **y** to this question, then messages triggered by a file system reaching the soft disk space limit will be sent to the user `root` of machine `leonardo`. You should answer **n** to this question if you, or the person responsible for ensuring free space in the file systems, rarely log in as `root` in this machine, and do not read the mail.

```
Enter notification address (e.g. 'joe@capitale'): sysadmin@main
```

This prompt appears only if you answer negatively to the last question. Enter the name and address of the person that should be notified when an audit file system

gets full.

```
Last chance to abort gracefully. Do you want to continue? [y|n]: n
```

If you have any doubts at all about the accuracy of your answers until now, you should abort the process by answering **n** to this question. If you do, you will see the message:

```
Aborting conversion.
Replies left in 'C2conv_input'. Script left in 'C2conv_script'.
Some additional file systems may now be mounted.
```

Your answers to `C2conv`'s prompts have been saved in the file `C2conv_input`. Taking this file as its input, the program `/usr/lib/c2convert` generates a shell script called `C2conv_script`. If you decide to abort at this point, these files are left for you to examine.

If you answer **y** to the above question and decide to continue, the script will be run and your answers will take effect. You will also be asked

```
Do you want to set a local password for 'audit'? [y|n]:
```

You should always be able to log in as user `audit`. This is the only user ID whose actions are never audited. It is the only user, therefore, who can do something if the system came to a halt because all auditing file systems are full. If your network does not run NIS, you should make sure that all servers and clients have a local user `audit` and a local password for it. If you are running NIS service, and you want an NIS password for `audit`, you must make sure that the local `/etc/passwd` file (in all machines except the NIS master server) has a line that says:

```
+audit::9:9:::
```

Notice the user ID and group ID of `audit`. Make sure that there are no collisions between them and other ID's that may exist from before your conversion to C2.

If you request the secure NFS option when `C2conv` asks you about it, then the password for the `audit` user on the client and the server of the file system *must* be the same. Because of this, when you request the NFS secure option, `C2conv` automatically modifies the clients' password entries for `audit` as in the example above, assuming that you will be using NIS to maintain the passwords in sync. On the other hand, because `C2conv` does not know whether you are answering questions from a NIS master server, it refrains from modifying the `audit` entry in the local `/etc/passwd`. If necessary, you should do it manually.

```
Some file systems that were not mounted when you started
may be mounted now.
```

This is `C2conv`'s last message. It is an advisory message to remind you that at the beginning of its run `C2conv` mounted all the local file systems from disk. If you want to return to your original single-user status, you must unmount them manually.

19.8. After Running C2conv

After you have successfully run `C2conv`, there are still a few tasks that you must perform by hand.

- If you are running NIS in your network, you should make sure that the `yppasswdd` daemon is run correctly, so that it changes `passwd.adjunct` instead of `passwd`. To ensure this, add the following line to the `/etc/rc.local` file in the NIS master:

```
/usr/etc/rpc.yppasswdd /pathname/passwd.adjunct -m
```

See the chapter on *The Network Information Service*, particularly the section that deals with changing the NIS password.

Also make sure that, as said above, there is no conflict between the user and group ID of `audit` and that of other users, and that the local `/etc/passwd` file has a `+` entry for the user `audit`.

- After you have brought up all the machines to multi-user mode, make sure that the audit daemon (`auditd`) is present in all of them. Enter:

```
machine# ps aux | grep auditd
```

If there is no audit daemon present in a given machine, make sure the following three lines are not commented out in the file `/etc/rc.local`:

```
if [ -f /usr/etc/auditd ]; then
    auditd;    (echo -n ' auditd') >/dev/console
fi
```

and boot the machine again.

- Once you have converted to C2 security, adding users is done as always. However, instead of editing only `/etc/passwd` and `/etc/group`, now you must also edit `/etc/security/passwd.adjunct` and `/etc/security/group.adjunct`.
- Adding machines to an existing C2 network varies in degree of difficulty depending on the type of machines. If you are adding a server and its clients, or a standalone, simply proceed as above, making sure that the constraints regarding the password for `audit` are observed. If you are adding a new client to a server in a network that has already been converted to C2, run `C2conv` on the client, make sure that the client can mount the audit file systems from the audit server(s), and reboot the client.
- If you want to add a new audit file system to the file systems currently available, you have to replicate by hand some of the actions of `C2conv`. If you are adding the device `/dev/xy0d`, for instance, on machine `addon`, follow these steps:
 1. Modify `addon`'s `fstab` file so the device is mounted on the directory `/etc/security/audit/addon`.

2. Modify all `fstab` files, both in file servers and clients, and have them NFS mount that file system in their `/etc/security/audit/addon` mount point. Use `mkdir` if the mount point does not exist.
3. Add a line that says
`dir:/etc/security/audit/addon/files` to the file `audit_control`.
4. Enter the command

```
# audit -s
```

to re-initialize the process.

- After running `C2conv`, you can verify that the process has completed correctly by making sure that the contents of the following files look correct in both servers and clients:

```
/etc/fstab
/etc/passwd
/etc/security/passwd.adjunct
/etc/group
/etc/security/group.adjunct
/etc/security/audit/audit_control
```

Suggestion: Do a full, recursive listing of `/etc/security/audit` to verify that the auditing mount points have been created correctly and, after rebooting, are mounted properly and with the right permissions. Type:

```
example# ls -lR /etc/security/audit
```

- The following is a sample run of a recursive `ls` on `/etc/security/audit`:

```

% ls -lR /etc/security/audit
total 4
drwx----- 4 audit          512 Dec 12 13:37 leonardo
drwx----- 4 audit          512 Dec 12 13:37 leonardo.1
-rw-r----- 1 audit          101 Dec 12 13:37 audit_control
-rw----- 1 audit           70 Dec 12 14:05 audit_data

/etc/security/audit/leonardo:
total 9
drwx----- 2 audit          512 Dec 12 14:05 files
drwxr-xr-x 2 root          8192 Dec 12 11:29 lost+found

/etc/security/audit/leonardo/files:
total 3
-rw----- 1 audit          658 Dec 12 14:02 19891212213824.19891212220255.leonardo
-rw----- 1 audit           92 Dec 12 14:05 19891212220544.19891212220549.leonardo
-rw----- 1 audit          180 Dec 12 14:05 19891212220549.not_terminated.leonardo

/etc/security/audit/leonardo/lost+found:
total 0

/etc/security/audit/leonardo.1:
total 9
drwx----- 2 audit          512 Dec 12 13:37 files
drwxr-xr-x 2 root          8192 Dec 12 11:29 lost+found

/etc/security/audit/leonardo.1/files:
total 0

/etc/security/audit/leonardo.1/lost+found:
total 0
%

```

- The following is a sample `/etc/security/audit/audit_control` file after running `C2conv`:

```

dir:/etc/security/audit/slapshot/files
dir:/etc/security/audit/slapshot.1/files
dir:/etc/security/audit/seraglio/files
dir:/etc/security/audit/seraglio.1/files
dir:/usr/tmp/files
flags:lo,p1
minfree:20

```

This file establishes four regular audit file systems and a temporary one. It specifies the system audit level as auditing all new logins and all unusual privileged operations. The `minfree` level is set at 20 percent.

19.9. Changing the Audit State

From time to time you may want to change what is being audited on a given machine or for a particular user. Suspicious activities may be occurring on a certain machine, but the user who is performing them might not be known. Or, a given user may warrant closer watching than before. Another consideration is that audit file systems may be filled with uninteresting information.

Changing the System Audit State

Changing the `flags`: line of `/etc/security/audit/audit_control` affects the system audit state. For instance, if the `flags`: line says:

```
flags:lo,p1
```

and you want to add auditing of regular privileged operations and administrative operations, you can edit the file and change the above line to

```
flags:lo,p1,ad,p0
```

Notice that there should be no spaces anywhere on this line.

After changing this file, you need to signal processes to reflect the changes you just made. As superuser, enter:

```
# audit -s
```

Changing the system audit state is fairly complicated internally. Since the user audit state overrides the system state, the audit daemon must calculate a new process audit state for every user and make the change to every process executing on the machine. On a busy machine with lots of users and a large number of processes, this can require a significant amount of time.

Changing the User Audit State

Remember that there are two possible kinds of changes to the user audit states: permanent (takes effect at login time) and immediate (takes effect after login time, and only for the duration of the present login session).

Permanent User Audit State

To affect a permanent change in a user's audit state you have to change the database that carries that information, that is, you have to change the appropriate entry in `/etc/security/passwd.adjunct`. The fields in this file are as follows:

Name	Password	min-label	max-label	default-label	always-audit	never-audit
User name	Encrypted password	Not used in C2			Items to audit at all times	Items to audit at no time

After running `C2conv`, for instance, the entry for the user `audit` looks like this:

```
audit:<encrypted password>::::all
```

The flag `all` in the seventh field (never audit) means that nothing is ever audited for this user. *Never change this flag.*

Supposing that the system audit state specifies `dr, da, lo, p0` (that is, auditing is carried out on data reads, data access, logins and regular privileged operations) and that

- you don't want to audit user `bob` in any of his successful operations,
- you want to see what kind of administrative operations he performs,

you would add `ad` as the sixth field of his entry in `passwd.adjunct` and `+dr, +da, +lo, +p0` as the seventh field of his entry. After these changes, Bob's entry in the `/etc/passwd.adjunct` file would look like this:

```
bob:<encrypted password>:::ad:+dr,+da,+lo,+po
```

After the changes are made to this file, the command:

```
# audit -d username
```

causes the file to be read and all processes owned by `username` are changed to reflect the changes. Note that all changes in the file are applied. Any previous immediate state changes are nullified.

Immediate User Audit State

The command:

```
# audit -u username state
```

changes the audit state for all processes with the audit user ID `username` to be the specified `state`. Note that this change affects all existing processes running for that user, but *not* any new login sessions by this user.

For example: You suspect that user `bob` has a program which runs `setuid root` that allows him to read other user's mail files. You execute the commands:

```
machine# su audit
audit% audit -u bob all
audit% cd /etc/security/audit
audit% tail +0f `sed 's/.*/:/' <audit_data` | praudit -s
```

The audit trail scrolls past, and when the suspicious activity occurs you can take appropriate actions.

Changing the Audit File

When the soft limit in a given file system is reached, auditing is switched to the next directory listed in `audit_control`. You can cause the switch to occur at any time. The command:

```
# audit -n
```


causes the audit daemon (`auditd`) to close the current audit file and start using a new audit file in the next available audit directory. The next available audit directory is listed in the audit daemon's internal directory list, which was formed the last time it read the `audit_control` file. To force the audit daemon to re-read the `audit_control` file, enter

```
# audit -s
```

19.10. Looking at the Audit Trail

The command `praudit` is the primary mechanism available for viewing the binary data contained in audit trails. Some hints on its usage are given here. The manual entry for `praudit(8)` describes the program more fully, and Appendix E describes the format of audit records.

Static Examination

With no parameters, `praudit` displays data in a manner suited for formal reports or documenting particular results. It is recommended that you use this form only for small audit trails.

When you use `praudit` with the `-l` option, it displays each audited event in a single line. This is handy for piping into `grep` or `awk`:

```
# praudit -l
```

A more compressed form of output results if the `-s` flag is supplied. This is generally useful only if the person reading the output is familiar with the short abbreviations used for audit classes.

For input into databases that expect user ID numbers instead of names, the `-r` option provides information in its numeric form where possible. This includes user IDs, group IDs, and audit class and record types.

Watching on the Fly

To watch the audit trail as events occur, issue the command:

```
audit% tail +0f <audit trail> | praudit -l -s
```

where *audit trail* is the name of the current audit trail file found in `/etc/security/audit/audit_data`. One convenient way to get the name of the audit file is to invoke the following commands:

```
machine# su audit
audit% cd /etc/security/audit
audit% tail +0f `sed 's/.*/:/' <audit_data>` | praudit -l
```

Remember that the user `audit` is not audited, so you cannot watch yourself if you are logged in as `audit`.

Stopping Auditing

You can stop the auditing at any time by entering:

```
machine# su audit
audit% audit -t
```

The above command kills the audit daemon `auditd` and stops the auditing.

19.11. When Audit File systems Are Full

If all audit file systems overflow, the best thing to do is to

- log in under the `audit` account
- make a backup of a single audit trail
- delete the backed up trail.

This should free up enough space so auditing can continue. Note that becoming superuser or logging in as `root` doesn't help, because superuser actions are audited, causing the system to hang until there is space on an audit file system somewhere.

19.12. Running `C2unconv`

Note: `C2unconv` does not affect auditing files or daemons.

You should run the `C2unconv` program if you want to *unconvert* from C2 password security back to the standard UNIX password scheme. `C2unconv` is a shell script that resides in `/usr/etc`. It modifies the following files:

```
/etc/passwd
/etc/group
/etc/security/passwd.adjunct
/etc/security/group.adjunct
```

`C2unconv` generates and runs a `sed` script that merges together the two split password files (into `/etc/passwd`) and the two split group files (into `/etc/group`).

Running `C2unconv` on a server only unconverts the server files. To unconvert the client files, log in to the client *before* running `C2unconv` on the server, and run `C2unconv`. Then halt the client(s) and run `C2unconv` on the server.

Note that `C2unconv` does only what it is supposed to do: It affects only the password and group files. You will still have all the other directories and files created by `C2conv`, as well as two extra NIS maps.

Administering Electronic Mail

SunOS electronic mail is handled by `sendmail`, a general internet network mail routing service. `sendmail` features aliasing and forwarding, automatic routing to network gateways, and flexible configuration.

This chapter explains how to install the mail system on your computer. The instructions are generic and should apply to most mail systems. However, if your mail system requires more specialized procedures, refer to the chapter on *Customizing sendmail Configuration Files*.

20.1. An Overview of the Electronic Mail System

The mail system consists of the following commands and files.

<code>/usr/ucb/mail</code>	UCB mail program, described in <code>mail(1)</code>
<code>/usr/bin/mailtool</code>	Window-based interface to the <code>sendmail</code> program.
<code>/usr/lib/sendmail</code>	Mail routing program
<code>/usr/lib/sendmail.mx</code>	Mail routing program linked with the domain name service resolver
<code>/etc/sendmail.cf</code>	Configuration file for mail routing
<code>/usr/lib/sendmail.main.cf</code>	Sample configuration file for main machines (see below)
<code>/usr/lib/sendmail.subsidiary.cf</code>	Sample configuration file for subsidiary machines (see below)
<code>/var/spool/mail</code>	Mail spooling directory for delivered mail
<code>/var/spool/mqueue</code>	Spool directory for mail going out over the network
<code>/var/spool/secretmail</code>	Secure mail directory
<code>/usr/bin/xsend</code>	Secure mail sender

<code>/usr/bin/xget</code>	Secure mail receiver
<code>/usr/bin/enroll</code>	To receive secure mail messages
<code>/etc/aliases</code>	Mail forwarding information
<code>/usr/ucb/newaliases</code>	Symbolic link to <code>/usr/lib/sendmail</code> .
<code>/usr/ucb/biff</code>	Mail notification enabler
<code>/usr/etc/in.comsat</code>	Mail notification daemon
<code>/usr/etc/syslogd</code>	Error message logger, used by <code>sendmail</code>

Here are the phases of mail collection and the programs involved.

Phase One: Message Collection

Processes Involved: `mailtool`, `/ucb/mail`

These processes provide an interface that collects messages. Users send mail by using the `Mail` command or `mailtool`, a user interface fully described in *SunOS User's Guide: Getting Started*, to edit the messages sent and received. (Refer to the `mail(1)` man page for more information about the `Mail` program.)

Phase Two: Message Routing

Processes Involved: `usr/lib/sendmail`

Message routing involves determining the route a message must take to get to the sender. `Mail` and `mailtool` pass messages from a sender to `sendmail` for routing. `sendmail` processes each piece of mail, using information in the configuration file `/etc/sendmail.cf` to determine network name syntax, aliasing, and forwarding information, and network topology. (Refer to the `sendmail(8)` man page for detailed information.)

Phase Three: Message Delivery

Processes involved: `/bin/mail`

Message delivery involves delivering the message to the appropriate machine. `sendmail` delivers the local mail by giving it to the program `/bin/mail`, `/bin/mail` adds the mail to the mailboxes in the directory `/var/spool/mail`, using a locking protocol to avoid problems with simultaneous updates. The file `/var/spool/mail/username` normally contains mail for each user name.

Phase Four: Message Retrieval

Processes involved: `mailtool`, `/ucb/mail`

Message retrieval involves the user accessing and displaying the message. After the mail is delivered, the local mail notification daemon `/usr/etc/in.comsat` notifies users who have the command `biff y` in their `.login` files that mail has arrived. The `mailtool` interface periodically checks users' `mbox` files and notifies them when mail has arrived.

Only you can read your mail file. However, anyone with the superuser password can read others' files, including their mail. To send mail that is secure against

any possible perusal (except by a code-breaker), use the secret mail facility, which encrypts the mail so that no one can read it. The man page `xsend(1)` fully describes this facility. Note that `xsend` does not work over the network.

Domain Names and sendmail

By default, `sendmail` accepts as mail addresses the Internet standard domain names for network domains. These standards make it possible for any Internet system in the world to send or receive mail with any other Internet system. This is why each network domain within the Internet must have a unique name. Remember that domains are administrative divisions only. They have nothing to do with the way in which hosts on the network are connected. Typically, all machines in a given domain are connected to each other, but this is only a convenience to simplify administration.

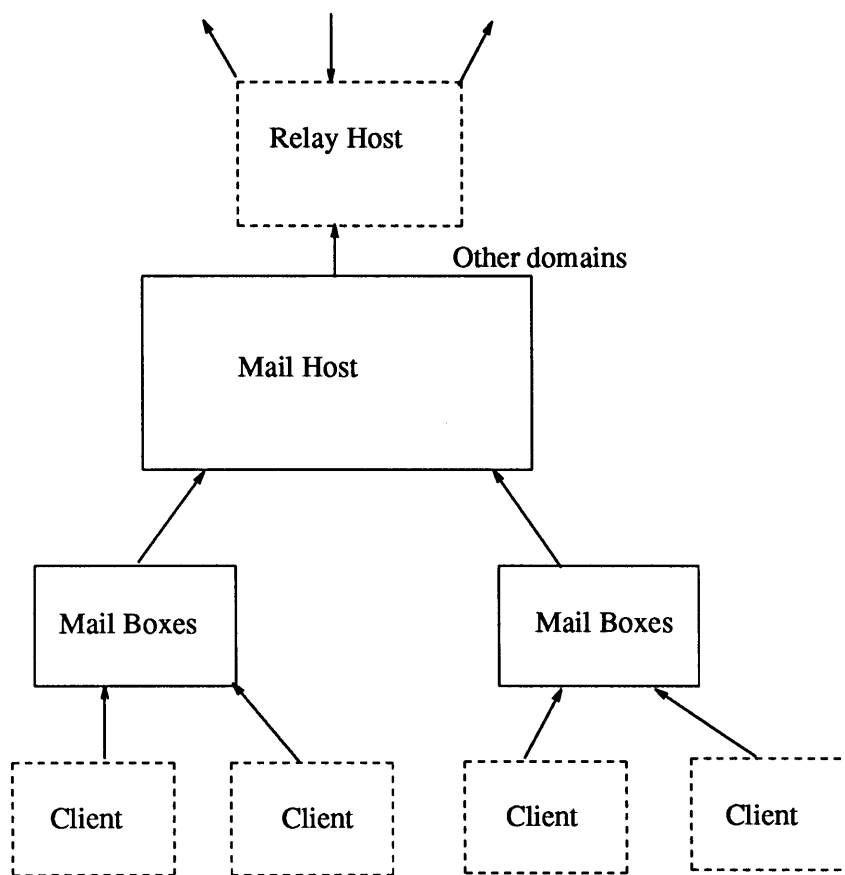
Internet names are divided into domains and subdomains, as is described in detail in the chapter *Administering Domain Name Service*. There are only a small number of top-level domains, for example `.COM` and `.EDU`. In some countries, the name of the country is the top level domain. For example, some systems in the United Kingdom use the top level domain `.UK`. In the United States, the top level domain `.US` is used for some personal systems.

In a mail address, domains nest inside one another like SunOS directory pathnames, except that the names go from right to left. Thus `joe.dance.com` is the name of host `joe` in subdomain `dance`, which is in domain `.com`, in the same way that `/etc/hosts.equiv` is file `hosts.equiv` in directory `/etc`.

If you have not done so already, you should register your domain with the Network Information Center (NIC) at SRI International, as described in the chapter *The SunOS Network Environment*.

20.2. Configuring Machines for Mail Operation

From a `sendmail` perspective, three types of machines exist in a network: at least one mailbox server, mail clients, and a mail host. Figure 20-1 illustrates their relationship to one another.

Figure 20-1 *A Typical Electronic Mail Configuration*

This section defines these machines and explains how to configure them.

Setting Up an NFS Mailbox Server and Its Clients

A *mailbox server* is any machine that actually stores mailboxes in the directory `/var/spool/mail`. In Release 4.1, client machines can mount their mailboxes through NFS from a mailbox server. This enables users to log in to any machine, including the server, and read their mail. You can designate an NFS server as a mailbox server by having it export `/var/spool/mail`. However, other types of machines, even diskless clients, can operate as mailbox servers.

`/var/spool/mail` holds the individual mailboxes for users on the network. Each file called `/var/spool/mail/user_name` is an individual mailbox, containing all the (non-secret) mail for a particular user.

Any NFS server can be a mailbox server, including machines from companies other than Sun or machines running earlier releases of the SunOS operating system. To set up a machine as an NFS mailbox server, you need to edit `/etc/exports`, so that the server exports `/var/spool/mail`.

The mailbox server is responsible for routing all mail from a client. When a client sends mail, the mailbox server puts it in a queue for delivery. Thereafter, the client can reboot or even power down, yet its mail will safely reach the recipient—barring other network problems, of course. When the recipient gets

the client's mail, the path in the message's "From:" line contains the name of the mailbox server. If the recipient chooses to respond, the response goes to the user's mailbox in `/var/spool/mail` on the mailbox server, not directly to the client.

Converting Clients to Use Mailbox Servers

The following instructions explain how to set up the mailbox server's clients.

1. Make sure clients are halted so that no mail is lost during the conversion.
2. Log in as superuser on the mailbox server.
3. If any mailboxes exist on the client, move them to the mailbox server's mailbox directory. Type the following.

```
# mv /export/root/client_name/var/spool/mail/* /var/spool/mail
```

4. Edit `fstab` in each client's `/export/root/client_name/etc` directory, adding an entry with the following format.

```
mailbox_server:/var/spool/mail /var/spool/mail nfs,rw
```

where `mailbox_server` is the server's name.

5. When you are finished editing their `/etc/fstab` files reboot each client to have the new `/etc/fstab` file take affect.

Once you have performed these tasks, mail operations can proceed. The mailbox server receives incoming mail from `sendmail`. The `/bin/mail` program running on the server puts the mail in the appropriate mailbox in `/var/spool/mail`. Each client mounts its mailbox through the `/var/spool/mail` entry in its `/etc/fstab` file.

Setting Up the Mailbox Server Alias

The next step in setting up mailbox servers is to assign all aliases to the mailbox server. You do this by editing the `/etc/aliases` file. (The later section, *Setting Up the Postmaster Alias*, fully describes `/etc/aliases`.)

Suppose machine "ballet" is the mailbox server for a particular organization or group. The existing `/etc/aliases` file on the network might resemble the following.

```
root: sysadmin@ballet
shamira: shamira@raks
benny: benny@samba
```

where "ballet" is the mailbox server name and "raks" and "samba" are client names.

To set up the mailbox server alias for server ballet, you would first log in to ballet. Then change the client names to the mailbox server names, so that `/etc/aliases` might look like this.

```
root: sysadmin@ballet
shamira: shamira@ballet
benny: benny@ballet
```

Be very careful that all aliases in the file resolve to names on the mailbox server, not the clients. In other words, if you leave the entry for user shamira as

```
shamira: shamira@raks
```

and do not change it to

```
shamira: shamira@ballet
```

shamira's outgoing mail will take an extra trip over the network between raks and ballet.

Setting Up the Mailhost and Subsidiary Machines

The next step in mail configuration is to define which machine on your network is the main mail machine, or *mailhost*, and which machines are mail subsidiaries. Subsidiary mail machines directly distribute mail to recipients in the same domain. However, when a mail subsidiary encounters mail destined for a machine in a different domain, the subsidiary forwards it to the mailhost for routing.

The mailhost must have the file `/usr/lib/sendmail.main.cf` installed as `/etc/sendmail.cf` in its root file system. Subsidiary mail machines have the file `/usr/lib/sendmail.subsidiary.cf` installed as `/etc/sendmail.cf` in their root file systems. The mailhost/subsidiary configuration simplifies administration by reducing the number of machines with custom mail configuration files. In most cases, you will find the default `/usr/lib/sendmail.main.cf` and `/usr/lib/sendmail.subsidiary.cf` appropriate for your network. However, you may want to make a few simple changes to customize the `sendmail.cf` files for your particular network. Should you want to tailor these configuration files for your network, refer to *Customizing sendmail Configuration Files* for more information.

A good candidate for mailhost is a machine attached to an Ethernet and to phone lines, or a machine configured as a router to the Internet. Note that if you have a non-networked standalone in a time-sharing configuration, you can nevertheless establish electronic mail for the configuration. Simply treat the standalone as the mailhost of a one machine network. Similarly, if you have several machines on an Ethernet and none have phones, pick one as the mailhost and leave the others as subsidiaries. You might connect your network to other domains in the future, perhaps by using the Sunlink MHS product.

Configuring Subsidiary Machines

SunInstall installs the default subsidiary configuration file, `/usr/lib/sendmail.subsidiary.cf` on each machine. Thus, you do not have to do anything more to configure a subsidiary machine, unless you want to use a configuration file other than the default. Refer to *Customizing sendmail Configuration Files* for specific instructions. Then put the subsidiary configuration file in `/export/root/proto.local/etc/sendmail.cf`.

You can make simple changes to the subsidiary configuration file to fit the needs of your particular network. For example, you can change the way in which the network domain name appears in mail messages.

The primary domain name is the name that will be applied to all headers generated by `sendmail`. By default, the visible part of the network domain name (as described in *The SunOS Network Environment*) is the primary domain name. There are no aliases for the primary domain name. If you want to change the domain name that appears in the mail headers generated by `sendmail`, you need to add the appropriate lines to `sendmail.cf`.

In this file, the line beginning with the letters “Dm” sets the primary domain name, which appears on outgoing mail. You can configure `sendmail` to receive mail for your domain that is addressed to an aliased domain name, rather than the primary domain name. To do so, you use the lines in `sendmail.cf` beginning with “Cm.” On each line beginning with Cm, you can add aliases for the local domain. `sendmail` automatically adds the primary domain to this list of aliases.

Note: The following instructions also apply when you are configuring the mailhost.

Suppose your network has the domain name `hq.dance.com`. Its displayed domain name on mail messages would be `dance.com`. To have `sendmail` accept different domain names for incoming mail, do the following:

1. Log in as superuser on the subsidiary machine.
2. Edit `sendmail.cf`.
3. Find the line beginning with Cm. It should have the primary domain name, `dance.com`, plus any aliases by which your domain can be accessed by the outside world.
4. Add any aliases for your domain that you want `sendmail` to know about, for example:

```
Cmdance.com dance.uucp
```

Thereafter, `sendmail` will recognize that incoming mail sent to either of these domain names should in fact be sent to the local domain.

On a subsidiary machine with phone lines, you can edit the `/etc/sendmail.cf` file so that `sendmail` routes mail received from `uucp` to certain hosts via the local phone lines. This is more efficient than having all `uucp` traffic go through the mailhost. (The chapter *Administering the UUCP System* explains how to do this.) Follow these procedures.

1. Log in as superuser on the subsidiary machine with a phone line.
2. Edit `sendmail.cf` and find the following line.

```
# local UUCP connections -- not forwarded to mailhost
CV
```

3. Put the names of the local `uucp` sites on the end of the CV line, or create additional CV lines.

For example, you could do the following.

```
CV rome prussia georgia
```

This allows only this host to send mail messages to rome, prussia, and georgia via uucp.

Configuring the Mailhost

The first step in creating the mailhost is to have the file `/usr/lib/sendmail.main.cf` become the mail configuration file for that machine. Follow this procedure:

1. Log in as superuser on the machine to become mailhost.
2. Type the following.

```
# cp /usr/lib/sendmail.main.cf /etc/sendmail.cf
```

Thereafter, `sendmail` will read `/etc/sendmail.cf` and recognize that this machine is the mailhost.

3. Create an entry for your new mailhost in the `/etc/hosts` file on the master NIS server, or on all hosts on a network without NIS. The entry should have this form.

```
IP address mailhost_name mailhost
```

For example, suppose you've selected the machine `ballet` as the mailhost. Its entry in `/etc/hosts` should resemble the example below.

```
129.255.99.99 ballet mailhost
```

You can optionally edit `/etc/sendmail.cf` on the mailhost to suit your particular network.

Establishing the Relay Host

Your mailhost may have a uucp or Ethernet connection with a machine called a *relay host* that will relay mail to you. The relay host handles unresolved mail—mail with an address for which `sendmail` couldn't find a recipient in your domain. If one exists, `sendmail` uses the relay host for sending and receiving mail outside your network domain.

Depending on your domain's needs, you can configure the mailhost as the relay host, or configure another machine as relay host. In fact, you may choose to configure more than one relay host for your domain. For example, if you have uucp connections, you should configure the machine with uucp connections as the relay host.

The relay host runs at least one mail-related protocol called a *mailer*. Each mailer specifies a policy and the mechanics to use when delivering mail. The mailer on the sending relay host must match the mailer on the receiving machine.

sendmail provides for several different kinds of mailers. Each mailer is defined by a *rule set*, as explained in *Customizing sendmail Configuration Files*. The sample `sendmail.cf` file defines several available mailers, such as `smartuucp`, `ddn`, `ether`, and `uucp`. You can define others.

Once you have established a relay host for your domain, you need to edit `sendmail.cf`. Look for the following block of lines in `sendmail.cf`.

```
# major relay mailer
DMsmartuucp

# major relay host
DRddn-gateway
CRddn-gateway
```

Note that `DM` is the primary name for your network domain. Change the line `DMsmartuucp` to the name of the mailer that your mailhost uses to connect to the major relay host. For example, specify `uucp` if your relay host uses `uucp` connections to the outside world, or `ddn` if the relay host is on the Internet.

Change the name following the letters `DR` to the name of the relay host. Change the name following the letters `CR` to the name of the relay host, followed by any aliases that machine may be known as.

For example, if your local network includes a machine called “`cmu-cs-vlsi`” that is on the Internet, you might use the following entry.

```
# major relay mailer
DMddn

#major relay host
DRcmu-cs-vlsi
CRcmu-cs-vlsi
```

On the other hand, your relay host might be `uucp` host `ucbvax`, which means you might use the following entry.

```
# major relay mailer
DMuucp

#major relay host
DRucbvax
CRucbvax
```

This change enables you to mail to an address such as “`charlie@MIT.EDU`.” Even though your mailhost may not be on the Internet, the message will arrive. If you are using Sunlink/MHS, refer to your installation manual for more information.

Setting Up the Postmaster Alias

Someone at your site, possibly yourself, should have the responsibility for handling problems with electronic mail. This person should have the alias *Postmaster*, a title which is recognized throughout the electronic mail community. For example, you can send mail to `postmaster` at other network sites if you have problems with mail originating at those sites.

The `/etc/aliases` File

The `/etc/aliases` file on a local machine contains all names by which a machine or person is known; `sendmail` reads it to determine mailing addresses. It is completely described in the man page `aliases(5)`. You can use uppercase letters in names to the left of the colon in `/etc/aliases`. However, it is much safer to only use lowercase letters. `/etc/aliases` has local aliases for individuals and groups, and special aliases. A local alias has this form.

```
name: address[, address]
```

where *name* is the person or group's name and *address* is the address of a recipient in the group. A typical local alias might be:

```
benny:benny@samba
```

A special alias has this form.

```
owner-aliasname: address
```

`postmaster` is a special alias. Every local `/etc/aliases` file should have a `postmaster` entry; here is the default entry.

```
# Following alias is required by the mail protocol, RFC 822
# Set it to the address of a HUMAN who deals with this system's mail problems
postmaster: root
```

To establish the `postmaster` alias, change "root" to the user name of the person who will act as postmaster. Then messages directed to "postmaster" arrive with that person's other mail. If this postmaster also manages a domain with more than one host, add the `postmaster` alias to `/etc/aliases` on all hosts, or, for networks running NIS, in the `mail.aliases` map on the NIS master server.

If you manage the mail system for several domains, change `/etc/aliases` on all of them to forward `postmaster` mail to the machine where the postmaster usually reads mail. Suppose you are user `amina`, the network postmaster, and you use machine `raks`. You would use the following entry for `postmaster` in `/etc/aliases`.

```
postmaster: amina@raks
```

As postmaster, you may not want to have users' mail mixed in with your personal mail. Here are procedures that help you avoid this by redirecting `postmaster` mail into a separate file on your machine.

1. Log in as superuser on each mail client. (If your network runs NIS, you have to perform only Steps 1 and 2 on the master NIS server.)
2. Add an alias such as the following.

```
postmaster: sysadmin@raks
```

to each mail client's `/etc/aliases`. This entry tells `sendmail` to direct mail to the `postmaster` alias to `sysadmin` on machine `raks`.

3. Log in as superuser on your own machine.
4. Add your own local mail alias to `/etc/aliases`. For example, `postmaster amina` would add the following entry to `/etc/aliases` on her own machine, `raks`.

```
sysadmin: /home/amina/mailadm
```

This entry defines an alias for `sysadmin`: the file `/home/amina/mailadm`.

5. Exit superuser and log in with your own user name.
6. Type the following to create the file `mailadm`.

```
% touch /home/amina/mailadm
% chmod og+w /home/amina/mailadm
```

Conversely, you might want `amina` to own the file but exclude others from any type of access. In this case, you would type this command.

```
% chmod 6600 /home/amina/mailadm
```

7. Type the following to have the file read.

```
% mail -f mailadm
```

You can also create aliases for people or groups of people called *mailing lists* when setting up the `postmaster` alias. The actual `/etc/aliases` file contains instructions for doing this.

Each time you edit `/etc/aliases`, you must run the `newaliases` program to rebuild the local alias database. If your network runs NIS, run `make in /var/yp` on the master NIS server after updating domain wide aliases.

Handling Undelivered Mail

By default, any time a message is returned as undeliverable by `sendmail`, a copy of the message header is sent to the `postmaster`. You can optionally disable this feature by editing the `sendmail.cf` file. Look for the following lines.

```
# cc my postmaster on error replies I generate
OPPostmaster
```

Change them as shown.

```
# cc my postmaster on error replies I generate
OPnobody
```

As shown above, you might want to create a separate file for undelivered mail, instead of mixing it with your personal mail.

Special Considerations for Networks with NIS

Networks running NIS have additional steps you must take and additional services you can add to sendmail service. Most importantly, after you have finished modifying `/etc/hosts` and `/etc/aliases` on the NIS master server, remember to propagate the NIS maps associated with these files. Type the following.

```
# cd /var/yp
# make
```

You can now use `mail.aliases` for domain-wide mail aliases. Thereafter, you should not have to remember hostnames when sending mail. `mail.aliases` will usually contain a copy of all the aliases known to a central mail machine.

Using Inverted NIS Domain Names

The inversion of the NIS domain-wide aliases can be used to simplify mail going outside the current domain. For example, consider the following `/etc/aliases` entries.

```
benny:benny@samba
gkelly:kelly@jazz
```

Note: On networks with NIS, you do not have to explicitly state a host-name. `sendmail` gets this information from the `mail.aliases` map.

When user `kelly` sends mail to user `benny`, the mail header displays the sender name, `kelly@jazz`. However, if `kelly` sends a message to a user called `placido@song.com`, the sender name on the message will be `gkelly@dance.com`. “`dance.com`” is the visible part of the domain name.

For incoming mail, `sendmail` normally replaces the left side of an alias entry with the right side. Thus the left side of the alias `gkelly` is replaced by the right side of the alias, `kelly@jazz`.

`sendmail` performs inverted alias processing on outgoing mail, replacing the right side of the alias with the left. This prevents the specific mailbox servers on a network from being visible outside the local domain. Therefore, if user `kelly` decides to switch to using host `jazz` as a mailbox server, user `placido@song.com` can still reply to `gkelly@dance.com`. The mail will automatically go to the right mailbox server.

Mail and the Internet Domain Name Server

In a domain running NIS, the NIS maps `hosts.byname` and `host.byaddr` resolve host names and addresses. You may wish to use the Internet domain name service for machines directly connected to the Internet. Consider doing this only if you can route between your machine and one or more of the “official” root servers.

Even if you are on one of these networks, you should still keep the `sendmail.cf` files on all clients and mailbox servers with standard configurations. You only need to customize the mailhost `sendmail.cf` to take advantage of domain name service.

Install `/usr/lib/sendmail.mx` in place of `/usr/lib/sendmail` on the mailhost to use domain name service. Each machine running `sendmail.mx` must have either `/etc/resolv.conf` or `/etc/named.boot` set up properly to allow name resolution or at least a caching server. Refer to the chapter *Administering Domain Name Service* and the `resolv.conf(5)` man page for more information.

Testing Your Mail Configuration

First, reboot all systems whose configuration files you have changed. Then, send test messages from various machines on the network. To do so, go to a particular machine and type the following.

```
samba% /usr/lib/sendmail -v </dev/null names
```

This command sends a null message to the specified recipient name, and displays messages while it runs. Try the following tests:

- Send mail to yourself or other people on the local machine by addressing the message in the command above to a regular user name, such as `root`.
- If you are on an Ethernet, send mail to someone on another machine, as in `root@jazz`. Try this in three directions: from the main machine to a subsidiary machine, vice versa, and from a subsidiary machine to another subsidiary machine, if more than two are on the network.
- If you have a relay host, send some mail to another domain from the mailhost. This ensures that the relay mailer and host are selected properly.
- If you have set up a `uucp` connection on your phone line to another host, you can send mail to someone at that host and they can send mail back, or call you on the phone if they receive it. If the connection is bi-directional, you can test it by entering this information.

```
% mail their_host!your_host!your_name < anyfile
```

Try having them send mail to you. For example, you could send to `ucbvax!azhar` if you have a connection to `ucbvax`. `sendmail` will not be able to tell you whether the message really got through, since it hands the message to `uucp` for delivery. You have to ask the human at the other end if mail has been received. To get an idea of the message’s progress, type the following.

```
% '% uulog'
```

or

```
% tail -f /usr/spool/uucp/.Log/uucico
```

For more information, refer to *Administering the UUCP System*.

- Send a message to “postmaster” on various machines and make sure that it comes to your postmaster’s mailbox, so that when other sites send you mail as postmaster, you will see it.

Diagnosing Problems with Mail Delivery

Here are some tools you can use for diagnosing mail problems.

The `/usr/lib/sendmail` command has a number of options for troubleshooting problems. For example, you can type the following.

```
% /usr/lib/sendmail -v -bv recipient
```

to verify the aliases and the deliverability of a given *recipient*. Here is an example of the resulting display from this command.

```
% /usr/lib/sendmail -v -bv shamira@raks
shamira... aliased to mwong
mwong... aliased to shamira@raks
shamira@raks... deliverable
```

`sendmail` also includes a test mode invoked by the `bt` option, as shown below.

```
% /usr/lib/sendmail -bt
```

Here are instructions for using test mode.

1. Invoke test mode by typing the following.

```
% /usr/lib/sendmail -bt
ADDRESS TEST MODE
Enter <ruleset> <address>
```

2. Respond to the last prompt by typing a zero, then the mail address you want to test, for example, `benny@samba`.

```
> 0 benny@samba
rewrite: ruleset 3 input: "benny" "@" "samba"
rewrite: ruleset 6 input: "benny" "<" "@" "samba" ">"
.
.
.
```

The diagnostic information that `sendmail` displays is fully described in *Customizing sendmail Configuration Files*.

The `mconnect` program is another diagnostic tool that you can use to open connections to other `sendmail` systems over the network. `mconnect` runs interactively, so that you can issue it various diagnostic commands. For example, the `VERFY` command of `SMTP` (the protocol used to deliver mail over the network) performs the same operation as the `-bv` option to `sendmail`, and `VERB` invokes verbose mode like the `-v` option.

Other diagnostic tools include:

- Received lines in the header of the message.

These trace which systems the message was relayed through. Note that in the `uucp` network, many sites do not update these lines, and that in the Internet, the lines often get rearranged. You can straighten them out by looking at the date and time in each line. Don't forget to account for different time zones.

- Messages from "MAILER-DAEMON."

These typically report delivery problems.

- The system log, for delivery problems in your group of workstations.

`sendmail` always records what it is doing in the system log, as explained in the next subsection. You might want to modify the `crontab` file to run a shell script nightly that searches the log for `SYSERR` messages and mails any that it finds to postmaster. In this way, problems are often fixed before anyone notices them, and the mail system runs more smoothly.

The System Log

The system log is supported by the `syslogd` program, fully described in the chapter *Administering Workstations*, and in the man page `syslogd(8)`. Just as you define a machine called "mailhost" to handle mail relaying, you can define a machine called "loghost" in `/etc/hosts` to hold all the logs for an entire NIS domain.

Format

Each line in the system log consists of a timestamp, the name of the machine that generated it (for logging from several machines over the Ethernet), and a message.

Levels

`syslog` can log a large amount of information. The log is arranged as succession of levels. At the lowest level, only unusual occurrences are logged. At the highest level, even the most mundane and uninteresting events are recorded for posterity. As a convention, log levels under ten are considered "useful;" log levels above ten are usually for debugging purposes.

Administering the UUCP System

21.1. Introduction

The UUCP system lets computers using the UNIX operating system communicate with each other and with remote terminals. In this chapter we use the term “UUCP” to denote the whole range of files and utilities that comprise the system, of which the program `uucp` is only a part. These utilities range from those used to copy files between computers (`uucp` and `uuto`) to those used for remote login and command execution (`cu` and `uux`).

As an administrator, you need to be familiar with the administrative tools, logs, and database files used by the UUCP system.

The new version of UUCP distributed with the SunOS 4.1 operating system is based on the version distributed with AT&T’s System V Release 3, which in turn is based on Honey DanBer UUCP. It also has some elements from BSD 4.3, as well as some Sun modifications.

If you are installing UUCP for the first time in your system, you should feel free to go directly to the next section. If you are changing from a previous version of UUCP to this one, you should read the section at the end of this chapter that presents some pointers for converting to the present one.

21.2. UUCP Hardware

Before your computer can communicate with other computers, you must set up the hardware to complete the communications link. The cables and other hardware you will need depend on how you want to connect the computers: direct links, telephone lines, or network.

Direct Links

You can create a direct link to another computer by running cables between serial ports on the two computers. Direct links are useful where two computers communicate regularly and are physically close—within 50 feet of each other. You can use a limited distance modem to increase this distance somewhat. Transfer rates of up to 19200 bits per second (bps) are possible when computers are directly linked.

Telephone Lines

Using an Automatic Call Unit (ACU) or modem your computer can communicate with other computers over standard phone lines. The ACU dials the telephone number requested by UUCP. The computer it is trying to contact must have a modem capable of answering incoming calls.

This chapter assumes that, if you are going to use UUCP over a telephone line, you have read the chapter *Adding Hardware to Your System*, you have already connected a modem to your machine, and are able to use `tip` or `cu` to dial out.

Network

UUCP can also communicate over the network. Once your computer is established as a node on a network, it will be able to contact any other computer connected to the network.

21.3. UUCP Programs

The UUCP programs can be divided into two categories: user programs and administrative programs. The following paragraphs describe the programs in each category.

User Programs

The UUCP user programs are in `/usr/bin`. No special permission is needed to use these programs. These commands are all described in the *SunOS Reference Manual*.

`cu` Connects your computer to a remote computer so you can be logged in on both at the same time, allowing you to transfer files or execute commands on either computer without dropping the initial link.

`uucp`

Lets a user copy a file from one computer to another. It creates work files and data files, queues the job for transfer, and calls the `uucico` daemon, which in turn attempts to contact the remote computer.

`uuto`

Copies files from one computer to a public spool directory on another computer (`/var/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`.

`uupick`

Retrieves the files placed under `/var/spool/uucppublic/receive` when files are transferred to a computer using `uuto`.

`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.

`uustat`

Displays the status of requested transfers (`uucp`, `uuto`, or `uux`). It also provides you with a means of controlling queued transfers.

Administrative Programs

Most of the administrative programs are in `/usr/lib/uucp`. Most of the basic data base files and shell scripts are in `/etc/uucp`. The only exception is `uulog`, which is in `/usr/bin`. These commands are described in the *SunOS Reference Manual*.

You should use the `uucp` login ID when you administer the UUCP system because it owns the programs and spooled data files. The home directory of the `uucp` login ID is `/var/spool/uucppublic`.

`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`.

`uucleanup`

Cleans up the spool directory. It is normally executed from a shell script called `uudemon.cleanup`, which is started by `cron`.

`Utry`

Tests call processing capabilities and does a moderate amount of debugging. It invokes the `uucico` daemon to establish a communication link between your computer and the remote computer you specify.

`uuccheck`

Checks for the presence of UUCP directories, programs, and support files. It can also check certain parts of the `Permissions` file for obvious syntactic errors.

Daemons

There are three daemons in the UUCP system. A daemon is a program that runs as a background process and performs a system-wide public function. These daemons handle file transfers and command executions. They can also be run manually from the shell.

`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 execute files, logs results, and 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.

The `uucico` daemon is executed by `uucp`, `uuto`, and `uux` programs, after all the required files have been created, to contact the remote computer. It is also executed by the `uusched` and `Utry` programs.

`uuxqt`

Executes remote execution requests. It searches the spool directory for execute files (always named `X.file`) that have been 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. If the files are present and can be accessed, `uuxqt` checks the `Permissions` file to verify that it has permission to execute the requested command. The `uuxqt` daemon is executed by the

`uudemon.hour` shell script, which is started by `cron`.

`uusched`

Schedules the queued work in the spool directory. Before starting the `uucico` daemon, `uusched` randomizes the order in which remote computers will be called. `uusched` is initially run at boot time by `/etc/rc`, and is subsequently executed by a shell script called `uudemon.hour`, which is started by `cron`.

21.4. Starting UUCP

UUCP comes with four shell scripts that will poll remote machines, reschedule transmissions, and clean up old log files and unsuccessful transmissions. These shell scripts should be executed regularly to keep UUCP running smoothly. The following crontab file for `uucp` is created at installation time in `/usr/lib/uucp/uudemon.crontab`:

```
#
#ident "@(#)uudemon.crontab 1.1 88/05/16"
#
48 8,12,16 * * * /etc/uucp/uudemon.admin
45 23 * * * /etc/uucp/uudemon.cleanup
0 * * * * /etc/uucp/uudemon.poll
11,41 * * * * /etc/uucp/uudemon.hour
```

To activate it, become superuser and enter:

```
# su uucp
# crontab < /usr/lib/uucp/uudemon.crontab
```

Normally, these scripts are run automatically by `cron`, though they can also be run manually.

`uudemon.poll`

The `uudemon.poll` shell script, as delivered, does the following:

- Reads the `Poll` file (`/etc/uucp/Poll`) once an hour.
- If any of the machines in the `Poll` file are scheduled to be polled, a work file (`C.sysnxxxx`) is placed in the `/var/spool/uucp/nodename` directory, where `nodename` is replaced by the name of the machine.

The shell script is scheduled to run once an hour, before `uudemon.hour`, so that the work files will be there when `uudemon.hour` is called.

`uudemon.hour`

The `uudemon.hour` shell script you receive with your machine does the following:

- Calls the `uusched` program to search the spool directories for work files (`C.`) that have not been processed and schedules these files for transfer to a remote machine.

- Calls the `uuxqt` daemon to search the spool directories for execute files (X.) that have been transferred to your computer and were not processed at the time they were transferred.

As delivered, this is run twice an hour. You may want it to run more often if you expect high failure rates.

`uudemon.admin`

The `uudemon.admin` shell script, as delivered, does the following:

- Runs the `uustat` command with `-p` and `-q` options. The `-q` reports on the status of work files (C.), data files (D.), and execute files (X.) that are queued. The `-p` prints process information for networking processes listed in the lock files (`/var/spool/locks`).
- Sends resulting status information to the `uucp` administrative login via mail.

`uudemon.cleanup`

The delivered `uudemon.cleanup` shell script does the following:

- Takes log files for individual machines from the `/var/spool/uucp/.Log` directory, merges them, and places them in the `/var/spool/uucp/.Old` directory with other old log information. If log files get large, the `ulimit` may need to be increased.
- Removes work files (C.) 7-days old or older, data files (D.) 7 days old or older, and execute files (X.) two days old or older from the spool files.
- Returns to the sender mail that cannot be delivered.
- Mails a summary of the status information gathered during the current day to the UUCP administrative login (`uucp`).

21.5. Supporting Data Base

The UUCP system support files are in the `/etc/uucp` directory. The descriptions below provide details on the structure of these files so you can edit them manually.

Devices

Contains information concerning the location and line speed of the automatic call unit, direct links, and network devices.

Dialers

Contains character strings required to negotiate with modems (automatic calling devices) in the establishment of connections to remote computers.

Systems

Contains information needed by the `uucico` daemon and the `cu` program to establish a link to a remote computer. It contains information such as the name of the remote computer, the name of the connecting device associated with the remote computer, when the computer can be reached, telephone number, login ID, and password.

Dialcodes

This file contains dial-code abbreviations that may be used in the phone number field of `Systems` file entries.

Permissions

This file defines the level of access that is granted to computers when they attempt to transfer files or remotely execute commands on your computer.

Poll

This file defines computers that are to be polled by your system and when they are polled.

Sysfiles

This file is used to assign different or multiple files to be used by uucico and cu as Systems, Devices, and Dialers files.

There are several other files that may be considered part of the supporting data base, but are not directly related to the process of establishing a link and transferring files. These files—Maxuuxqts, Maxuuscheds, and remote.unknown—are described later on in the section *Other UUCP Files*.

21.6. Devices File

The Devices file (/etc/uucp/Devices) contains information for all the devices that may be used to establish a link to a remote computer, devices such as automatic call units, direct links, and network connections.

Each entry in the Devices file has the following format:

Table 21-1 *Format and example of Devices File*

Format	Type	Line	Line2	Class	Dialer-Token-Pairs
Example	ACU	cua0	-	1200	hayes \D

Each of these fields is defined below:

Note: 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.

Type

This field may contain one of two keywords (Direct or ACU), the name of a port selector, or a system name.

Direct

This keyword indicates a Direct Link to another computer or a port selector. This is for cu connections only. A separate entry should be made for each direct dialout line that you want to reference through the -l option of cu.

ACU

This keyword indicates that the link to a remote computer (whether through cu or through UUCP) is made through an automatic call unit (Automatic Dial Modem). This modem may be connected either directly to your computer or indirectly through a port selector.

port selector

This value can be replaced by the name of a port selector. micom and develcon are the only ones for which there are caller scripts in the Dialers file. You can add your own port selector entries to the Dialers file.

Sys-Name

This value indicates a direct link to a particular computer. (*Sys-Name* is replaced by the name of the computer.) This naming scheme is used to convey the fact that the line associated with this *Devices* entry is for a particular computer in the *Systems* file.

The keyword used in the *Type* field is matched against the third field of *Systems* file entries as shown below:

File	Entry						
Devices	ACU	cua0	-	1200	penril		
Systems	eagle	Any	ACU	1200	NY3251	ogin: nuucp	ssword: Oakgrass

You can designate a protocol to use for a device within this field. See the *Protocols* section at the end of the description of this file.

Line

This field contains the device name of the line (port) associated with the *Devices* entry. For instance, if the Automatic Dial Modem for a particular entry was attached to the `/dev/cua0` line, the name entered in this field would be `cua0`.

Line2

This is a placeholder. Always use a hyphen (-) here.[†]

Class

If the keyword `ACU` or `Direct` is used in the *Type* field, *Class* may be just the speed of the device. However, it may contain a letter and a speed (for example, `C1200`, `D1200`) to differentiate between classes of dialers (for example, 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. In such a case, it becomes necessary to distinguish which line(s) should be used for internal communications and which should be used for external communications. The keyword used in the *Class* field of the *Devices* file is matched against the fourth field of *Systems* file entries as shown below:

File	Entry						
Devices	ACU	cua0	-	1200	penril		
Systems	eagle	Any	ACU	1200	NY3251	ogin: nuucp	ssword: Oakgrass

Some devices can be used at any speed, so the keyword `Any` may be used in the *Class* field. If `Any` is used, the line will match any speed requested in a

[†] In systems that support the 801 type dialer (SunOS does not) if the keyword `ACU` was used in the *Type* field and the `ACU` is an 801 type dialer, *Line2* would contain the device name of the 801 dialer. (801 type `ACU`s do not contain a modem. Therefore, a separate modem is required and would have to be connected to a different line, defined in the *Line* field.) This means that one line would be allocated to the modem and another to the dialer. Since non-801 dialers do not use this configuration, the *Line2* field is ignored by them, but it must still contain a hyphen (-) as a placeholder.

Systems file entry. If this field is Any and the Systems file *Class* field is Any, the speed defaults to 1200 bps.

Dialer-Token-Pairs:

This field contains pairs of dialers and tokens. The *dialer* portion may be the name of an automatic dial modem, a port selector, or it may be *direct* for a Direct Link device. You can have any number of Dialer-Token-Pairs. The *token* portion may be supplied immediately following the *dialer* portion or if not present, it will be taken from a related entry in the Systems file.

This field has the format:

dialer token dialer token

where the last pair may or may not be present, depending on the associated device (*dialer*). In most cases, the last pair contains only a *dialer* portion and the *token* portion is retrieved from the *Phone* field of the Systems file entry.

A valid entry in the *dialer* portion may be defined in the *Dialers* file or may be the special dialer type *TCP*, which is compiled into the software and is therefore available without an entry for it in the *Dialers* file.

See below, under *Protocols* and under *Running UUCP over TCP/IP*.

The *Dialer-Token-Pairs (DTP)* field may be structured four different ways, depending on the device associated with the entry:

1. If an automatic dialing modem is connected directly to a port on your computer, the *DTP* field of the associated *Devices* file entry will only have one pair. This pair would normally be the name of the modem. This name is used to match the particular *Devices* file entry with an entry in the *Dialers* file. Therefore, the *dialer* field must match the first field of a *Dialers* file entry as shown below:

File	Entry						
Devices	ACU	cua0	-	1200	ventel		
Dialers	ventel	=&-%	""	\r\p\r\c	\$	<K\T%%\r>\c	ONLINE!

Notice that only the *dialer* portion (*ventel*) is present in the *DTP* 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. (NT is implied, see below.) Backslash sequences are described below.

2. If a direct link is established to a particular computer, the *DTP* field of the associated entry would contain the keyword *direct*. This is true for both types of direct link entries, *Direct* and *System-Name* (refer to discussion on the *Type* field).
3. If a computer with which you wish to communicate is on the same port selector switch as your computer, your computer must first access the switch

and the switch can make the connection to the other computer. In this type of entry, there is only one pair. The *dialer* portion is used to match a Dialers file entry as shown below:

File	Entry						
Devices	develcon	cua0	-	1200	develcon	\D	
Dialers	develcon	""	""	\pr\ps\c	est:\007	\E\D\e	\007

As shown, the *token* portion is left blank (the \D is *not* a token, it is part of the Dialer field), which indicates that it is retrieved from the Systems file. The Systems file entry for this particular computer will contain the token in the *Phone* field, which is normally reserved for the phone number of the computer (refer to Systems file, *Phone* field). This type of *DTP* contains an escape character (\D), which ensures that the contents of the *Phone* field will not be interpreted as a valid entry in the Dialcodes file.

4. If an automatic dialing modem is connected to a switch, your computer must first access the switch and the switch will make the connection to the automatic dialing modem. This type of entry requires two *dialer-token-pairs*. The *dialer* portion of each pair (fifth and seventh fields of entry) will be used to match entries in the Dialers file as shown below:

File	Entry						
Devices	ACU	cua0	-	1200	develcon	vent	ventel
Dialers	develcon	""	""	\pr\ps\c	est:\007	\E\D\e	\007
Dialers	ventel	=&-%	""	\r\p\r\c	\$	<K\T%%\r>\c	ONLINE!

In the first pair, *develcon* is the dialer and *vent* is the token that is passed to the Develcon switch to tell it which device (*ventel* modem) to connect to your computer. This token would be unique for each port selector since each switch may be set up differently. Once the *ventel* modem has been connected, the second pair is accessed, where *ventel* is the dialer and the token is retrieved from the Systems file.

There are two escape characters that may appear in a *DTP* field:

- \T Indicates that the *Phone (token)* field should be translated 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*, etc.). Therefore, the translation will not take place until the caller script is accessed.
- \D Indicates that the *Phone (token)* field should not be translated using the Dialcodes file. If no escape character is specified at the end of a Devices entry, the \D is assumed (default). A \D is also used in the Dialers file with entries associated with network switches (*develcon* and *micom*).

Protocols

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 (see `Systems` file, `type` field). If you do specify the protocol, you must use the form `Type,Protocol [Protocol]` (for example, `TCP,te`). Available protocols are:

- g This protocol is slower and more reliable than `e`. It is good for transmission over noisy telephone lines.
- e This protocol is faster than `g`, but it assumes transmission over error-free channels that are message oriented (as opposed to byte stream oriented like TCP/IP) between System V systems.
- f This protocol is used for transmission over X.25 connections. It relies on flow control of the data stream, and it is meant for working over links that can (almost) be guaranteed to be error-free, specifically X.25/PAD links. A checksum is carried out over a whole file only. If a transport fails the receiver can request retransmission(s).
- t This protocol is used for transmissions over TCP/IP and other reliable connections. It assumes error-free transmissions and is commonly used over TCP/IP to talk to 4.3BSD systems.

Here is an example of adding a protocol designation to a device entry:

```
TCP,te -- Any TCP --
```

This says, for device `TCP`, try to use the `t` protocol. If the other end refuses, use the `e` protocol.

Note that neither `e` nor `t` are appropriate for use over modems, since even if the modem assures error-free transmission, data can still be dropped between the modem and the cpu.

21.7. Dialers File

The `Dialers` file (`/etc/uucp/Dialers`) specifies the initial conversation that must take place on a line before it can be made available for transferring data. This conversation is usually a sequence of ASCII strings that is transmitted and expected, and it is often used to dial a phone number using an ASCII dialer. It is also known as a *chat script*.

As shown in the examples in the preceding section, the fifth field in a `Devices` file entry is an index into the `Dialers` file or a special dialer type (TCP). Here an attempt is made to match the fifth field in the `Devices` file with the first field of each `Dialers` file entry. In addition, each odd numbered `Devices` field starting with the seventh position is used as an index into the `Dialers` file. If the match succeeds, the `Dialers` entry is interpreted to perform the dialer negotiations. Each entry in the `Dialers` file has the following format:

Table 21-2 *Format and example of Dialers File*

Format	Dialer	Substitutions	Expect-Send
Example	ventel	=&-%	"" \r\p\r\c \$ <K\T%%\r>\c ONLINE!

The *dialer* field matches the fifth and additional odd numbered fields in the *Devices* file. The *substitutions* field is a translate string: the first of each pair of characters is mapped to the second character in the pair. This is usually used to translate = and - into whatever the dialer requires for "wait for dialtone" and "pause."

The remaining *expect-send* fields are character strings. Below are some sample entries distributed with the UUCP system in the *Dialers* file.

```
penril =W-P "" \d > Q\c : \d- > s\p9\c )-W\p\r\ds\p9\c-) y\c : \E\TP > 9\c ok
ventel =&-% "" \r\p\r\c $ <K\T%%\r>\c ONLINE!
vadic =K-K "" \005\p *- \005\p-* \005\p-* D\p BER? \E\T\%e \r\c LINE
develcon "" "" \pr\ps\c est:\007 \E\D\%e \007
micom "" "" \s\c NAME? \D\r\c go
rixon =&-% "" \r\r\d $ s9\c )-W\r\ds9\c-) s\c : \T\r\c $ 9\c LINE
hayes =,-, "" \dA\pTE1V1X1Q0S2=255S12=255\r\c OK\r \EATDT\T\r\c CONNECT
tb1200 =W-, "" \dA\pA\pA\pTE1V1X1Q0S2=255S12=255S50=2\r\c OK\r \EATDT\T\r\c
CONNECT\s1200
tb2400 =W-, "" \dA\pA\pA\pTE1V1X1Q0S2=255S12=255S50=3\r\c OK\r \EATDT\T\r\c
CONNECT\s2400
tbfast =W-, "" \dA\pA\pA\pTE1V1X1Q0S2=255S12=255S50=255\r\c OK\r \EATDT\T\r\c
CONNECT\sFAST
att4000 =,-, "" ATZ\r\p\p OK\r ATZ\r OK\r\c \EATDT\T\r\c CONNECT
att4024 =+,-, "" atzod,o12=y,o4=n\r\c \006 atT\T\r\c ed
att2212c =+,-, "" atzod,o12=y,o4=n\r\c \006 atT\T\r\c ed
att2212C =+,-, "" atzod,o12=y,o4=n\r\c \006 atT\T\r\c ed
att2224b =+,-, "" atT\T\r\c ed
att2224B =+,-, "" atT\T\r\c ed
direct
```

The meaning of some of the escape characters (those beginning with "\") used in the *Dialers* file are listed below:

- \p pause (approximately ¼ to ½ second)
- \d delay (approximately 2 seconds)
- \D phone number or token without *Dialcodes* translation
- \T phone number or token with *Dialcodes* translation
- \K insert a BREAK
- \E enable echo checking (for slow devices)
- \e disable echo checking
- \r carriage return

`\c` no new-line or carriage return

`\n` send new-line

`\nnn`

send octal number. Additional escape characters that may be used are listed in the section discussing the `Systems` file.

The `penril` entry in the `Dialers` file is executed as follows:

```
penril =W-P "" \d > Q\c : \d- > s\p9\c )-W\p\r\ds\p9\c-) y\c : \E\TP > 9\c OK
```

This is interpreted as follows (where *E* means *expect*, and *S* means *send*):

dialer	substitution	E	S	E	S	E	S	E	S	E-S-E	S	E	S	E	S	E
penril	=W-P	""	\d	>	Q\c	:	\d-	>	s\p9\c)-W\p\r\ds\p9\c-)	y\c	:	\E\TP	>	9\c	OK

First, the substitution mechanism for the phone number argument is established, so that any `=` will be replaced with a `W` (wait for dialtone) and any `-` with a `P` (pause).

The handshake given by the remainder of the line works as follows:

`""` Wait for nothing. (In other words, proceed to the next thing.)

`\d` Delay for 2 seconds, then send a carriage-return.

`>` Wait for a `>`.

`Q\c`

Send a `Q` without a carriage-return.

`:` Expect a `:`.

`\d-`

Delay 2 seconds, send a `-` and a carriage-return.

`>` Wait for a `>`.

`s\p9\c`

Send an `s`, pause, send a `9` with no carriage-return.

`)-W\p\r\ds\p9\c-)`

Wait for a `)`. If it is not received, process the string between the `-` characters as follows. Send a `W`, pause, send a carriage-return, delay, send an `s`, pause, send a `9`, without a carriage-return, and then wait for the `)`.

`y\c`

Send a `y` with no carriage-return.

`:` Wait for a `:`.

`\E\TP`

Enable echo checking. (From this point on, whenever a character is transmitted, it will wait for the character to be received before doing anything else.) Then, send the phone number. The `\T` means take the phone

number passed as an argument and apply the `Dialcodes` translation and the modem function translation specified by field 2 of this entry. Then send a P and a carriage-return.

```
> Wait for a >.
9\c
  Send a 9 without a new-line.
OK Wait for the string OK.
```

Hardware Flow Control

You can also use the pseudo-send `STTY=value` string to set modem characteristics. For instance, `STTY=crtscts` enables hardware flow control.

`STTY` accepts all the stty modes. See `stty(1V)` and `termio(4)` in the *SunOS Reference Manual*.

The following example would enable hardware flow control in a `Dialers` entry:

```
dsi      =,-,      "" \dA\pTE1V1X5Q0S2=255S12=255*E1*F3*M1*S1\r\c OK\r \
\EATDT\T\r\c CONNECT\SEC STTY=crtscts
```

This pseudo-send string can also be used in entries in the `Systems` file.

Setting Parity

In some cases, you will have to reset the parity because the system that you are calling checks port parity and drops the line if it is wrong. The expect-send couplet `"" P_ZERO` sets parity to zero:

```
foo      =,-,      "" P_ZERO "" \dA\pTE1V1X1Q0S2=255S12=255\r\c OK\r \EATDT\T\r\c CONNECT
```

In the same manner, `P_EVEN` sets it to even (this is the default), `P_ODD` sets it to odd, and `P_ONE` sets it to one. This pseudo-send string can also be used in entries in the `Systems` file.

21.8. Systems File

The `Systems` file (`/etc/uucp/Systems`) contains the information needed by the `uucico` daemon to establish a communication link to a remote computer. Each entry in the file represents a computer that can be called by your computer. In addition, the UUCP software is configured by default to prevent any computer that does not appear in this file from logging in on your computer (refer to the section *Other UUCP Files* in this chapter for a description of the `remote.unknown` file). More than one entry may be present for a particular computer. The additional entries represent alternative communication paths that will be tried in sequential order.

Using the `Sysfiles`, you can define several files to be used as "Systems" files. See the description of the `Sysfiles` file for details.

Each entry in the `Systems` file has the following format:

Table 21-3 *Format and example of Systems File*

Format	<i>System-Name</i>	<i>Time</i>	<i>Type</i>	<i>Class</i>	<i>Phone</i>	<i>Login</i>
Example	eagle	Any	ACU	1200	5553251	ogin: nuucp ssword: Oakgrass

Each of these fields is defined in the following section.

System-name

This field contains the node name of the remote computer.

Time

This field is a string that indicates the day-of-week and time-of-day when the remote computer can be called. The format of the *Time* field is:

daytime[;retry]

The day portion may be a list containing some of the following:

Su Mo Tu We
for individual days

Wk for any week-day (Mo Tu We Th Fr)

Any
for any day

Never

for a passive arrangement with the remote computer. If the *Time* field is *Never*, your computer will never initiate a call to the remote computer. The call must be initiated by the remote computer. In other words, your computer is in a passive mode in respect to the remote computer (see discussion of `Permissions` file, `SENDFILES` option).

Here is an example:

Wk1700-0800, Sa, Su

This example allows calls from 5:00 p.m. to 8:00 am, Monday through Friday, and calls any time Saturday and Sunday. The example would be an effective way to call only when phone rates are low, if immediate transfer is not critical.

The *time* portion should be a range of times such as 0800-1230. If no *time* portion 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 a.m. and 8 a.m.

Retry

An 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, *Any; 9* is interpreted as call any time, but wait at least 9 minutes before retrying after a failure occurs. Note that changing the retry field before the 9 minutes have expired would have no effect, as from then on only the entry in `.Status/system` is relevant. In the absence of the *;retry* entry, an exponential backoff algorithm is used

(starting with a default wait time that grows larger as the number of failed attempts increases). The initial retry time is 5 minutes, and the maximum retry time is 23 hours. If `;time` is specified, that will always be the retry time. Otherwise, the backoff algorithm is used.

Grade

An optional subfield, *grade*, is available to specify the maximum grade at which data should be sent.

All UUCP requests are sorted by grade. A grade is denoted by a single letter or number, from 0 to 9, from a to z, and from A to Z, where 0 (zero) is the lowest and Z is the highest. A lower grade causes a job to be transmitted earlier during a particular conversation, that is, it is treated as more important. The default grade is N. Mail is typically sent at the default grade, while news is sent at grade d. A *max grade* specification prevents UUCP from sending data with a grade higher (that is, a priority lower) than the specified.

You can specify the grade in two ways:

- on the `uucico` command line with the option `-vgrade=X`, or
- in the *time* field of the `Systems` file, following a `'/`.

For instance,

```
eagle Any/9 ACU,g D1200 NY3251 ogin: nuucp \
      ssword: Oakgrass
```

would specify that requests for system `eagle` with a grade up to 9 are to be sent using this `Systems` file entry.

Type

This field contains the device type that should be used to establish the communication link to the remote computer. The keyword used in this field is matched against the first field of `Devices` file entries as shown below:

File	Entry						
Systems	eagle	Any	ACU, g	1200	NY3251	ogin: nuucp	ssword: Oakgrass
Devices	ACU	cua0	-	1200		penril	

You can define the protocol used to contact the system by adding it on to the *Type* field. The example above shows how to attach the protocol `g` to the device type `ACU`. See the information under the *Protocols* section in the description of the `Devices` file for details.

Class

This field is used to indicate the transfer speed of the device used in establishing the communication link. It may contain a letter and speed (for example, C1200, D1200) to differentiate between classes of dialers (refer to the discussion on the `Devices` file, *Class* field). Some devices can be used at any speed, so the keyword `Any` may be used. This field must match the *Class* field in the associated `Devices` file entry as shown below:

File	Entry						
Systems	eagle	Any	ACU	1200	NY3251	ogin: nuucp	sword: Oakgrass
Devices	ACU	cua0	-	1200		penril	

If information is not required for this field, use a - as a place holder for the field.

Phone

This field is used to provide the phone number (token) of the remote computer for automatic dialers (port selectors). The phone number is made up of an optional alphabetic abbreviation and a numeric part. If an abbreviation is used, it must be one that is listed in the `Dialcodes` file. For example:

File	Entry						
Systems	eagle	Any	ACU	1200	NY3251	ogin: nuucp	sword: Oakgrass
Dialcodes	NY	9=5551212					

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 4 seconds before dialing the next digit.

If your computer is connected to a port selector, you may access other computers that are connected to that selector. The `Systems` file entries for these computers will not have a phone number in the `Phone` field. Instead, this field will contain the token that must be passed on to the switch so it will know 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 ensure that this field is not translated using the `Dialcodes` file.

Login

This field contains login information given as a series of fields and subfields of the 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 may be made up of subfields of the form:

expect[-send-expect]...

where the *send* is sent if the prior *expect* is not successfully read, and the *expect* following the *send* is the next expected string. For example, with `login--login`, UUCP will expect to read the string `login`. If UUCP gets `login`, it will go on to the next field. If it does not get `login`, it will send nothing (followed by a carriage-return), then look for `login` again. If no characters are initially expected from the remote computer, the characters "" (null string) should be used in the first *expect* field. Note that all *send* fields will be sent followed by a carriage-return unless the *send* string is terminated with a \c.

Here is an example of a `Systems` file entry that uses an expect-send string:

```
owl Any ACU 1200 Chicago6013 "" \r ogin:-BREAK-ogin: \
uucpx word: xyzzz
```

This example says send a carriage return and wait for `ogin:` (for `Login:` or `login:`). If you don't get `ogin:`, send a `BREAK`. When you do get `ogin:` send the login name `uucpx`, then when you get `word:` (for `Password:` or `password:`) send the password `xyzzz`.

There are several escape characters that cause specific actions when they are a part of a string sent during the login sequence. The following escape characters are useful in UUCP communications:

- `\N` Send or expect a null character (ASCII NUL).
- `\b` Send or expect a backspace character.
- `\c` If at the end of a string, suppress the carriage-return that is normally sent. Ignored otherwise.
- `\d` Delay two seconds before sending or reading more characters.
- `\p` Pause for approximately $\frac{1}{4}$ to $\frac{1}{2}$ second.
- `\E` Start echo checking. (From this point on, whenever a character is transmitted, it will wait for the character to be received before doing anything else.)
- `\e` Echo check off.
- `\n` Send a new-line character.
- `\r` Send or expect a carriage-return.
- `\s` Send or expect a space character.
- `\t` Send or expect a tab character.
- `\\` Send or expect a `\` character.
- `EOT`
Send EOT followed by new-line twice.
- `BREAK`
Send a break character.
- `\K` Same as `BREAK`.
- `\ddd`
Collapse the octal digits (`ddd`) into a single character.

Hardware Flow Control

You can also use the pseudo-send `stty=value` string to set modem characteristics. For instance, `stty=crtsets` enables hardware flow control.

`stty` accepts all the `stty` modes. See `stty(1V)` and `termio(4)` in the *SunOS Reference Manual*.

The following example would enable hardware flow control in a `Systems` entry:

```
unix Any ACU 2400 12015551212 "" \r login:-\r-login:-\r-login: nuucp \
ssword: xxx "" STTY=crtsets
```

This pseudo-send string can also be used in entries in the `Dialers` file.

Setting Parity

In some cases, you will have to reset the parity because the system that you are calling checks port parity and drops the line if it is wrong. The expect-send couplet "" P_ZERO sets parity to zero if inserted at the beginning of a chat script. For example,

```
unix Any ACU 2400 12015551212 "" P_ZERO "" \r login:-\r-login:-\r-login: nuucp ssword: xxx
```

In the same manner, P_EVEN sets it to even (this is the default), P_ODD sets it to odd, and P_ONE sets it to one.

This pseudo-send string can also be used in entries in the `Dialers` file.

21.9. Dialcodes File

The `Dialcodes` file (`/etc/uucp/Dialcodes`) contains the dial-code abbreviations that can be used in the `Phone` field of the `Systems` file. Each entry has the format:

Table 21-4 *Format and example of Dialcodes File*

Format	<i>abb</i>	<i>dial-seq</i>
Example	NY	1=212

where *abb* is the abbreviation used in the `Systems` file `Phone` field and *dial-seq* is the dial sequence that is passed to the dialer when that particular `Systems` file entry is accessed.

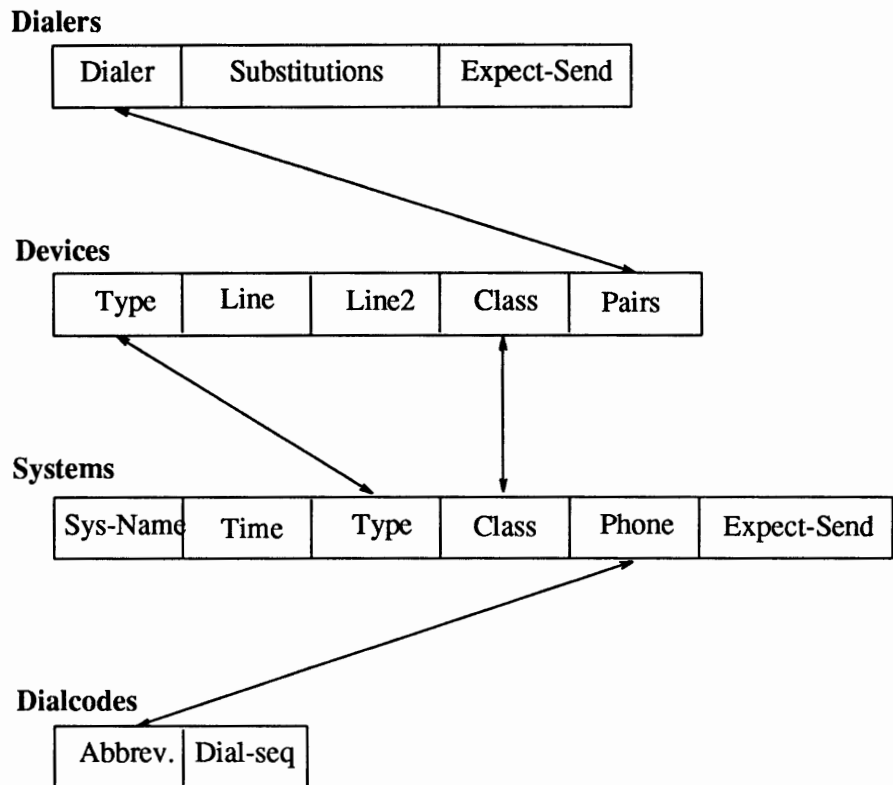
The entry

```
jt 9=847-
```

would be set up to work with a `Phone` field in the `Systems` file such as `jt7867`. When the entry containing `jt7867` is encountered, the sequence `9=847-7867` would be sent to the dialer if the token in the dialer-token-pair is `\T`.

Correspondences

Some of the main correspondences between the fields of the files `Devices`, `Dialers`, `Systems` and `Dialcodes`, as seen in the previous sections, can be summarized in the following figure:

Figure 21-1 *Correspondences Between the Files*

21.10. Permissions File

The `Permissions` file (`/etc/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. Another option is available that specifies the commands that a remote site can execute on the local computer.

How Entries are Structured

Each entry is a logical line with physical lines terminated by a `\` to indicate continuation. Entries are made up of options delimited by white space. Each option is a name/value pair in the following format:

```
name=value
```

Some *values* may be colon-separated lists. Note that no white space is allowed within an option assignment.

Comment lines begin with a `#` 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 logs in on

(calls) your computer.

Security Note: When a remote machine calls you, unless you have a unique login and password for that machine you don't know if the machine is who it says it is.

MACHINE

Specifies permissions that take effect when your computer logs in on (calls) a remote computer.

LOGNAME entries contain a LOGNAME option and MACHINE entries contain a MACHINE option. One entry may contain both options.

Considerations

The following items should be considered when using the `Permissions` file to restrict the level of access granted to remote computers:

- All login IDs used by remote computers to login for UUCP communications must appear in one and only one LOGNAME entry.
- Any site that is called whose name does not appear in a MACHINE entry, will have the following default permissions/restrictions:
 - Local send and receive requests will be executed.
 - The remote computer can send files to your computer's `/var/spool/uucppublic` directory.
 - The commands sent by the remote computer for execution on your computer must be one of the default commands; usually `rmail`.

REQUEST Option

When a remote computer calls your computer and requests to receive a file, this request can be granted or denied. The REQUEST option specifies whether the remote computer can request to set up file transfers from your computer. The string

```
REQUEST=yes
```

specifies that the remote computer can request to transfer files from your computer. The string

```
REQUEST=no
```

specifies that the remote computer cannot request to receive files from your computer. This is the default value. It will be 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 Option

When a remote computer calls your computer and completes its work, it may attempt to take work your computer has queued for it. The SENDFILES option specifies whether your computer can send the work queued for the remote computer.

The string

```
SENDFILES=yes
```

specifies that your computer may send the work that is queued for the remote computer as long as it logged in as one of the names in the LOGNAME option. This string is *mandatory* if your computer is in a "passive mode" with respect to the remote computer.

The string

```
SENDFILES=call
```

specifies that files queued in your computer will be sent only when your computer calls the remote computer. The `call` value is the default for the `SENDFILE` option. This option is only significant in `LOGNAME` entries since `MACHINE` entries apply when calls are made out to remote computers. If the option is used with a `MACHINE` entry, it will be ignored.

MYNAME Option

This option permits your computer to be known by a name different than the one returned by the command `hostname`. For instance, if you have unknowingly named your computer with the same name as that of some other system, or if you want your organization to be known as `widget` but all your modems are connected to another machine, `gadget`, you can have an entry in `gadget`'s `Permissions` file that says:

```
LOGNAME=Uworld MYNAME=widget
```

Now the system `world` will log in on the machine `gadget` thinking it is logging in on `widget`. In order for the other machine to know you also by this aliased name when you call it, you can have an entry that says:

```
MACHINE=world MYNAME=widget
```

You can also use the `MYNAME` option for testing purposes, since it allows your machine to call itself. However, since this option could be used to mask the real identity of a machine, you are strongly recommended to use the `VALIDATE` option described in this section.

READ and WRITE Options

These options specify the various parts of the file system that `uucico` can read from or write to. The `READ` and `WRITE` options can be used with either `MACHINE` or `LOGNAME` entries.

The default for both the `READ` and `WRITE` options is the `uucppublic` directory as shown in the following strings:

```
READ=/var/spool/uucppublic
WRITE=/var/spool/uucppublic
```

The strings

```
READ=/ WRITE=/
```

specify permission to access any file that can be accessed by a local user with "other" permissions.

The value of these entries is a colon separated list of path names. 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 path name of a file coming in or going out. To grant permission to deposit files in `/usr/news` as well as the public directory, the following values would be used with the `WRITE` option:

```
WRITE=/var/spool/uucppublic:/usr/news
```

It should be pointed out that if the `READ` and `WRITE` options are used, all path names must be specified because the path names are not added to the default list.

For instance, if the `/usr/news` path name was the only one specified in a `WRITE` option, permission to deposit files in the public directory would be denied.

You should be careful what directories you make accessible for reading and writing by remote systems. For example, you probably wouldn't want remote computers to be able to write over your `/etc/passwd` file so `/etc` shouldn't be open to writes.

NOREAD and NOWRITE Options

The `NOREAD` and `NOWRITE` options specify exceptions to the `READ` and `WRITE` options or defaults. The strings

```
READ=/ NOREAD=/etc WRITE=/var/spool/uucppublic
```

would permit reading any file except those in the `/etc` directory (and its subdirectories—remember, these are prefixes) and writing only to the default `/var/spool/uucppublic` directory. `NOWRITE` works in the same manner as the `NOREAD` option. The `NOREAD` and `NOWRITE` can be used in both `LOGNAME` and `MACHINE` entries.

CALLBACK Option

The `CALLBACK` option is used in `LOGNAME` entries to specify that no transaction will take 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 will be billed for the longer call.

The string

```
CALLBACK=yes
```

specifies that your computer must call the remote computer back before any file transfers will take place.

The default for the `CALLBACK` option is

```
CALLBACK=no
```

If you set `CALLBACK` to `yes`, then the permissions that affect the rest of the conversation must be specified in the `MACHINE` entry corresponding to the caller, not in the `LOGNAME`, as well as in the `LOGNAME` entry that the other machine may have set for yours.

Note that if two sites have the `CALLBACK` option set for each other, a conversation will never get started.

COMMANDS Option

The `COMMANDS` option can be hazardous to the security of your system. Use it with extreme care.

The `uux` program will generate remote execution requests and queue them to be transferred to the remote computer. Files and commands are sent to the target computer for remote execution. The `COMMANDS` option can be used in `MACHINE` entries to specify the commands that a remote computer can execute on your computer.

Note: This is an exception to the

rule that MACHINE entries apply only when your system calls out.

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 string

```
COMMANDS=rmail
```

indicates the default commands that a remote computer can execute on your computer. If a command string is used in a MACHINE entry, the default commands are overridden. For instance, the entry

```
MACHINE=owl:raven:hawk:dove \
COMMANDS=rmail:rnews:lp
```

overrides the COMMAND default so that the computers owl, raven, hawk, and dove can now execute rmail, rnews, and lp on your computer.

In addition to the names as specified above, there can be full path names of commands. For example,

```
COMMANDS=rmail:/usr/local/rnews:/usr/local/lp
```

specifies that command rmail uses the default search path. The default search path for UUCP is /bin and /usr/bin. When the remote computer specifies rnews or /usr/local/rnews for the command to be executed, /usr/local/rnews will be executed regardless of the default path. Likewise, /usr/local/lp is the lp command that will be executed.

Including the ALL value in the list means that any command from the remote computer(s) specified in the entry will be executed. If you use this value, you give the remote computer full access to your computer. **BE CAREFUL: This allows far more access than normal users have.** You should use this value only when both machines are at the same site, are closely connected, and the users are trusted.

The string

```
COMMANDS=/usr/local/rnews:ALL:/usr/local/lp
```

illustrates two points: The ALL value can appear anywhere in the string, and the path names specified for rnews and lp will be used (instead of the default) if the requested command does not contain the full path names for rnews or lp.

The VALIDATE option should be used with the COMMANDS option whenever potentially dangerous commands like cat and uucp are specified with the COMMANDS option. Any command that reads or writes files is potentially dangerous to local security when executed by the UUCP remote execution daemon (uuxqt).

VALIDATE Option

The VALIDATE option is used to provide a certain degree of verification of the caller's identity by cross-checking the host name of a calling machine against the login name it uses. The string

```
LOGNAME=Uwidget VALIDATE=widget:gadget
```

ensures that if any machine other than widget or gadget tries to log in as Uwidget, the connection is refused. The use of the VALIDATE option requires

that privileged computers have a unique login/password for UUCP transactions. An important aspect of this validation is that the login/password associated with this entry be protected. If an outsider gets that information, that particular VALIDATE option can no longer be considered secure.

The VALIDATE option is used in conjunction with the COMMANDS option when specifying commands that are potentially dangerous to your computer's security. (VALIDATE is merely an added level of security on top of 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 LOGNAME entry

```
LOGNAME=uucpfriend VALIDATE=eagle:owl:hawk
```

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`. As can be seen, if an outsider gets the `uucpfriend` login/password, masquerading is trivial.

But what does this have to do with the COMMANDS option, which only appears in MACHINE entries? It links 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 computer is logged in. In fact, it is an asynchronous process with no knowledge of what computer sent the execution request. Therefore, the real question is how does your computer 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 the 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, or if the computer name does not appear in the Permissions file, the default list will be used.

The following example shows the relationship between the MACHINE and LOGNAME entries:

```
MACHINE=eagle:owl:hawk REQUEST=yes \
COMMANDS=rmail:/usr/local/rnews \
READ=/ WRITE=/

LOGNAME=uucpz VALIDATE=eagle:owl:hawk \
REQUEST=yes SENDFILES=yes \
READ=/ WRITE=/
```

The value in the COMMANDS option means that remote mail and

`/usr/local/rnews` can be executed by remote users.

In the first entry, you must make the assumption that when you want to call one of the computers listed, you are really calling either `eagle`, `owl`, or `hawk`. Therefore, 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 will also be put in the privileged spool directory. You therefore have to validate that the computer has the privileged login `uucpz`.

MACHINE Entry for "OTHER"

You may want to specify different option values for the computers your computer calls that are not mentioned in specific MACHINE entries. This may occur when there are many computers calling in, and the command set changes from time to time. The name "OTHER" for the computer name is used for this entry as shown below:

```
MACHINE=OTHER \
COMMANDS=rmail:rnews:/usr/local/Photo:/usr/local/xp
```

All other options available for the MACHINE entry may also be set for the computers that are not mentioned in other MACHINE entries.

Combining MACHINE and LOGNAME

It is possible to combine MACHINE and LOGNAME entries into a single entry where the common options are the same. For example, the two entries

```
MACHINE=eagle:owl:hawk REQUEST=yes \
READ=/ WRITE=/

LOGNAME=uucpz REQUEST=yes SENDFILES=yes \
READ=/ WRITE=/
```

share the same REQUEST, READ, and WRITE options. These two entries can be merged as shown below:

```
MACHINE=eagle:owl:hawk REQUEST=yes \
LOGNAME=uucpz SENDFILES=yes \
READ=/ WRITE=/
```

Forwarding

When sending files through a series of machines, the intermediary ones must have the command `uucp` among their COMMANDS options. That is, if you enter the command

```
% uucp filename oak\!willow\!pine\!/usr/spool/uucppublic
```

then the only way this forwarding operation will work is if machine `willow` permits `oak` to execute the program `uucp`, and if `oak` permits your machine the same. The machine `pine`, being the last in this particular chain, does not have to permit the command `uucp`. Machines are not normally set up this way.

21.11. Poll File

The `Poll` file (`/etc/uucp/Poll`) contains information for polling remote computers. Each entry in the `Poll` file contains the name of a remote computer to call, followed by a `TAB` character (a space won't work), and finally the hours the computer should be called. The format of entries in the `Poll` file are:

```
sys-name<TAB>hour ...
```

For example the entry:

```
eagle 0 4 8 12 16 20
```

will provide polling of computer `eagle` every four hours.

The `uudemon.poll` script does not actually perform the poll. It merely sets up a polling work file (always named `C.file`), in the `spool` directory that will be seen by the scheduler, which is started by `uudemon.hour`.

21.12. Sysfiles File

The `/etc/uucp/Sysfiles` file lets you assign different files to be used by `uucp` and `cu` as `Systems`, `Devices`, and `Dialers` files. Here are some cases where this optional file may be useful.

- You may want different `Systems` files so requests for login services can be made to different addresses than `uucp` services.
- You may want different `Dialers` files to use different handshaking for `cu` and `uucp`.
- You may want to have multiple `Systems`, `Dialers`, and `Devices` files. The `Systems` file in particular may become large, making it more convenient to split it into several smaller files. The format of the `Sysfiles` file is

```
service=w systems=x:x dialers=y:y devices=z:z
```

where `w` is replaced by `uucico`, `cu`, or both separated by a colon; `x` is one or more files to be used as the `Systems` file, with each file name separated by a colon and read in the order presented; `y` is one or more files to be used as the `Dialers` file; and `z` is one or more files to be used as the `Devices` file. Each file is assumed to be relative to the `/etc/uucp` directory, unless a full path is given. A backslash (`\`) can be used to continue an entry on to the next line.

Here's an example of using a local `Systems` file in addition to the usual `Systems` file:

```
service=uucico:cu systems=Systems:Local_Systems
```

If this is in `/etc/uucp/Sysfiles`, then both `uucico` and `cu` will first look in `/etc/uucp/Systems`. If the system they are trying to call doesn't have an entry in that file, or if the entries in the file fail, then they'll look in

```
/etc/uucp/Local_Systems.
```

Given the above entry, `cu` and `uucico` will share the `Dialers` and `Devices` files.

When different `Systems` files are defined for `uucico` and `cu` services, your machine will store two different lists of `Systems`. You can print the `uucico` list using the `uname` command or the `cu` list using the `uname -c` command. The following is another example of the file, where the alternate files are consulted first and the default files are consulted in case of need.

```
service=uucico  systems=Systems.cico:Systems \
                dialers=Dialers.cico:Dialers \
                devices=Devices.cico:Devices
service=cu      systems=Systems.cu:Systems \
                dialers=Dialers.cu:Dialers \
                devices=Devices.cu:Devices
```

21.13. Other UUCP Files

There are three other files that impact the use of UUCP facilities. In most cases, the default values are fine and no changes are needed. If you want to change them, however, use any standard UNIX system text editor (`ed` or `vi`).

`Maxuuxqts`

This file defines the maximum number of `uuxqt` programs that can run at once. The default is 2.

`Maxuuscheds`

This file defines the maximum number of `uusched` programs that can run at once. The default is 2.

`remote.unknown`

This program executes when a machine that is not in any of the `Systems` starts a conversation. It will log the conversation attempt and fail to make a connection. If you change the permissions of this file so it cannot execute (`chmod 000 remote.unknown`), your system will accept any conversation requests. This is not a trivial change, and you should have very good reasons for doing it.

21.14. Administrative Files

The UUCP administrative files are described below. These files are created in spool directories to lock devices, hold temporary data, or keep information about remote transfers or executions.

`TM` (temporary data file)

These data files are created by UUCP processes under the spool directory (i.e., `/var/spool/uucp/X`) when a file is received from another computer. The directory `X` 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 path name specified in the `C.sysnxxx` file (discussed below) that caused the transmission. If processing is abnormally terminated, the `TM.pid.ddd` file may remain in the `X` directory. These files should be automatically removed by `uucleanup`.

LCK (lock file)

Lock files are created in the `/var/spool/locks` directory for each device in use. Lock files prevent duplicate conversations and multiple attempts to use the same calling device. The names of lock files have the format:

`LCK..str`

where `str` is either a device or computer name. These files may remain in the spool directory if the communications link is unexpectedly dropped (usually on computer crashes). The lock files will be ignored (removed) after the parent process is no longer active. The lock file contains the process ID of the process that created the lock.

c. (work file)

Work files are created in a spool directory when work (file transfers or remote command executions) has been queued for a remote computer. The names of work files have the format:

`C.sysnxxx`

where `sys` is the name of the remote computer, `n` is the ASCII character representing the grade (priority) of the work, and `xxxx` is the four digit job sequence number assigned by UUCP. Work files contain the following information:

- Full pathname of the file to be sent or requested
- Full pathname of the destination or user/file name
- User login name
- List of options
- Name of associated data file in the spool directory. If the `uucp -c` or `uuto -p` option was specified, a dummy name (`D.0`) is used
- Mode bits of the source file
- Remote user's login name to be notified upon completion of the transfer

D. (data file)

Data files are created when it is specified in the command line to copy the source file to the spool directory. The names of data files have the following format:

`D.systemxxxxyyy`

where `system` is the first five characters in the name of the remote computer, `xxxx` is a four-digit job sequence number assigned by `uucp`. The four digit job sequence number may be followed by a sub-sequence number, `yyy` that is used when there are several `D.` files created for a work (`C.`) file.

X. (execute file)

Execute files are created in the spool directory prior to remote command executions. The names of execute files have the following format:

`X.sysnxxxx`

where `sys` is the name of the remote computer, `n` is the character representing the grade (priority) of the work, and `xxxx` is a four digit sequence number assigned by UUCP. Execute files contain the following information:

- Requester's login and computer name.
- Name of file(s) required for execution.
- Input to be used as the standard input to the command string.
- Computer and file name to receive standard output from the command execution.
- Command string.
- Option lines for return status requests.

21.15. Running UUCP over TCP/IP

To run UUCP on top of TCP/IP, make sure that the following entry in `/etc/inetd.conf` is not commented out (that is, is not preceded by a # mark):

```
uucp  stream tcp  nowait  root    /usr/etc/in.uucpd  in.uucpd
```

In the case of TCP, entries in the Systems file are interpreted as having the following fields:

```
System-Name Time TCP Port networkname Standard-Login-Chat
```

A typical entry would look like this:

```
rochester Any tcp - ur-seneca ogin: Umachine ssword: xxx
```

Notice that the fifth field, `networkname`, permits you to specify explicitly the Internet host name. This is important for some sites; in the example above, the site may have a UUCP name like `rochester` but an Internet name like `ur-seneca`, and there may further be another Internet site called `rochester` that is a totally different machine.

Notice also that the Port field is listed as `-`. This is equivalent to listing it as `uucp`. The following entry in `/etc/services` sets up a port for UUCP:

```
uucp 540/tcp uucpd # uucp daemon
```

However, if you are running NIS, having an entry in the local `/etc/services` but not in the NIS map may not help.

The fourth field, `Port`, may be either a word or a number. If it is a word (`uucp` or `as-`) the port number will be obtained from the `services` database; if it is a number, that number will be used as the port. The following entry exemplifies this usage:

```
seismo Any TCP 33 seismo ogin: Umachine ssword: zzz
```

In almost every case, the `networkname` will be the same as the system name, and the port field will be `-` or `uucp`, which says to use the standard `uucp` port from the `services` file. The UUCP daemon (in `.uucpd`) expects the incoming site to send its login and password for authentication, and it prompts for them much as `getty` and `login do`.

21.16. Set Up Permissions File

The default `/etc/uucp/Permissions` file provides the maximum amount of security for your UUCP links. The file, as delivered, contains no entries.

You can set additional parameters for each machine to define:

- the ways it can receive files from your machine
- the directories it can read and write in
- the commands it can use for remote execution

A typical `Permissions` entry is:

```
MACHINE=datsun LOGNAME=Udatsun VALIDATE=datsun COMMANDS=rmail REQUEST=yes SENDFILES=yes
```

This allows files to be sent and received (to and from the “normal” UUCP directories, not from anywhere in the system) and causes the UUCP user name to be validated at login time.

21.17. Add uucp logins

For incoming UUCP (`uucico`) requests from remote machines to be handled properly, each machine has to have a login on your system.

An example of a typical entry in the `/etc/passwd` file is shown below.

```
Umachine:6L1bcFaRYh836:4:8:Machine:/var/spool/uucppublic:/usr/lib/uucp/uucico
```

By convention, the login name of a remote machine is the machine name preceded by a ‘U’. Note that the name should not exceed eight characters, so that in some cases truncation or abbreviation may be necessary.

This entry shows that a login request by `Umachine` is answered by `/usr/lib/uucp/uucico`. The home directory is `/var/spool/uucppublic`. The encrypted password is the result of using the command `passwd Umachine` as root. You must coordinate the password and the login name with the UUCP administrator of the remote machine. He or she, in turn, must add an appropriate entry, with login name and unencrypted password, in the remote machine’s `Systems` file.

Similarly, you must coordinate your machine's name and password with the UUCP administrators of all machines that you want to reach through UUCP.

21.18. Examples

The following are examples of the *minimal* entries required for two machines, called *caller* and *answer*, to establish some of the various types of connections: over TCP, over a direct line between two computers, over modems. In all cases, the examples illustrate the entries needed so that machine *caller* can call machine *answer* in order to copy a file into the directory `/usr/spool/uucppublic` of machine *answer*. You will have to modify most of the files mentioned in the examples, although in most cases you won't have to modify the `Dialers` file.

Once you have crossed the barrier of establishing one-way traffic from one machine to the other, modifying the basic entries to satisfy your site's requirements, as well as establishing two-way traffic, becomes much easier.

Connecting Through a Modem - LUUCP

File	Entry on <i>caller</i>	Entry on <i>answer</i>
Devices	ACU cua0 - 2400 hayes	No entry needed
Dialers	hayes =,-, "" \dA\pTE1V1X1Q0S2=255S12=255\r\c \ OK\r \EATDT\T\r\c CONNECT	No entry needed
Systems	answer Any ACU 2400 9=4155551212 \ in:--in: Ucaller word: abcdef	caller Never - - - -
Permissions	MACHINE= <i>answer</i>	LOGNAME= <i>Ucaller</i>
/etc/passwd	No entry needed	Entry for user <i>Ucaller</i>

Connecting Through TCP/IP

File	Entry on <i>caller</i>	Entry on <i>answer</i>
Devices	TCP,et - - Any TCP -	No entry needed
Systems	answer Any TCP - - in:--in: \ Ucaller word: abcdef	caller Never - - - -
Permissions	MACHINE= <i>answer</i>	LOGNAME= <i>Ucaller</i>
/etc/passwd	No entry needed	Entry for user <i>Ucaller</i>

Connecting Through a Direct Line - UUCP

File	Entry on <i>caller</i>	Entry on <i>answer</i>
Devices	<code>answer cua0 - 19200 direct</code>	No entry needed
Systems	<code>answer Any answer 19200 - in:—in: \ Ucaller word: abcdef</code>	<code>caller Never - - - -</code>
Permissions	<code>MACHINE=answer</code>	<code>LOGNAME=Ucaller</code>
/etc/passwd	No entry needed	Entry for user <i>Ucaller</i>

Connecting Through a Direct Line - cu

File	Entry on <i>caller</i>	Entry on <i>answer</i>
Devices	<code>Direct cua0 - 2400 direct</code>	No entry needed
Systems	No entry needed	No entry needed
Permissions	No entry needed	No entry needed
/etc/passwd	No entry needed	Entry for user calling in

Connecting Through a Modem - cu

File	Entry on <i>caller</i>	Entry on <i>answer</i>
Devices	<code>ACU cua0 - 2400 hayes</code>	No entry needed
Systems	No entry needed	No entry needed
Permissions	No entry needed	No entry needed
/etc/passwd	No entry needed	Entry for user calling in

21.19. UUCP Maintenance

UUCP does not require much maintenance. Apart from making sure that the cron-tab file is in place, as described in the section *Starting UUCP*, all you have to worry about is the growth of mail files and the public directory.

Mail for UUCP

All mail messages generated by the UUCP programs and scripts go to the user `uucp`. If you do not log in frequently as that user, you may not be aware of the mail that is accumulating (and consuming disk space). To solve this, make an alias in `/etc/aliases` and redirect that mail either to `root` or to yourself and other persons responsible for the maintenance of UUCP. Don't forget to run the `newaliases` command after modifying the `aliases` file.

The Public Directory

The directory `/var/spool/uucppublic` is the one place in every system to and from which UUCP is guaranteed by default to be able to copy files. Every user can `cd` into it, read and/or write files in it. However, its sticky bit is set, so its mode is `01777`. Users, therefore, will not be able to remove files that have been copied to it and that belong to `uucp`. Only the system administrator, logged in as `root` or as `uucp`, can remove files from this directory. To prevent the uncontrolled accumulation of files in this directory, you should make sure to clean it up periodically.

If this is inconvenient for users, encourage them to use `uuto` and `uupick` (see the man pages) rather than removing the sticky bit, which is there for security reasons. You can also make the mode of the directory more restrictive, perhaps just to a group of people, or, if you do not want to run the risk of having someone fill your disk, you can even deny UUCP access to it.

21.20. UUCP Debugging

These procedures describe how to go about solving common problems that may be encountered with UUCP.

Check for Faulty ACU/Modem

You can check if the automatic call units or modems are not working properly in several ways.

- Run `uustat -q`. This will give counts and reasons for contact failure.
- Run `cu -d -lline`. This will let you call over a particular line and print debugging information on the attempt. The line must be defined as Direct in the devices file. (You must add a telephone number to the end of the command line if the line is connected to an autodialer or the device must be set up as `direct`.)

Check Systems File

Check that you have up-to-date information in your systems file if you are having trouble contacting a particular machine. Some things that may be out of date for a machine are its:

- Phone number
- Login
- Password

Debug Transmissions

If you are unable to contact a particular machine, you can check out communications to that machine with `Uutry` and `uucp`.

Step 1:

To simply try to make contact, run:

```
% /usr/lib/uucp/Uutry -r machine
```

where *machine* is replaced with the node name of the machine you are having problems contacting. This command will:

- Start the transfer daemon (`uucico`) with debugging. You will get more debugging information if you are `root`.
- Direct the debugging output to `/tmp/machine`,
- Print the debugging output to your terminal (`tail -f`). Hit `CONTROL-C` to end output.

You can copy the output from `/tmp/machine` if you want to save it.

Step 2:

If `Uutry` doesn't isolate the problem, try to queue a job by running:

```
% uucp -r file machine\!/dir/file
```

where *file* is replaced by the file you want to transfer, *machine* is replaced by the machine you want to copy to, and *dir/file* is where the file will be placed on the other machine. The `-r` option will queue a job but not start the transfer.

Now use `Uutry` again. If you still cannot solve the problem, you may need to call support personnel. Save the debugging output; it will help diagnose the problem.

You may also want to decrease or increase the level of debugging provided by `Uutry` through the `-x#` option, where `#` indicates the debug level. The default debug level for `Uutry` is 5. A level of 3 will provide basic information as to when and how the connection is established, but not much information about the transmission itself. A debug level of 9, on the other hand, provides exhaustive information about the transmission process. Be aware that debugging occurs at both ends of the transmission, so if you intend to use a level higher than 5 on a moderately large text you should have the courtesy of getting in touch with the administrator of the other site and agreeing on a good time for doing this.

Check Error Messages

There are two types of error messages for UUCP: ASSERT and STATUS.

ASSERT Error Messages

When a process is aborted, ASSERT error messages are recorded in `/var/spool/uucp/.Admin/errors`. These messages include the file name, `sccsid`, line number, and text. These messages usually result from system problems.

STATUS Error Messages

Status error messages are stored in the `/var/spool/uucp/.Status` directory. The directory contains a separate file for each remote machine your computer attempts to communicate with. These files contain status information on the attempted communication and whether it was successful.

- Check Basic Information** There are several commands you can use to check for basic networking information.
- `uuname`
Use this command to list those machines your machine can contact.
- `uulog`
Use this command to display the contents of the log directories for particular hosts.
- `uucheck -v`
Run this command to check for the presence of files and directories needed by `uucp`. This command also checks the `Permissions` file and outputs information on the permissions you have set up.
- 21.21. This Version of UUCP** The major differences between the present version of UUCP and past versions are: different file organization and file names, increased security, more protocols and speedier spooling.
- Data Files** The old `L.sys`, `L-devices`, `L-dialcodes`, `L.cmds`, `SEQF` and `USERFILE` have been replaced and expanded by the files `Systems`, `Devices`, `Dialers`, `Dialcodes` and `Permissions` in the directory `/etc/uucp`. Of particular note is the file `Permissions`, which now has a wealth of options that permit the establishment of fairly secure connections to the outside world.
- Log and Status Files** Instead of one file for logging all messages (`LOGFILE`) there is now one file per system per command: `.Log/uucico/system`, `.Log/uucp/system`, `.Log/uux/system` and `.Log/uuxqt/system` (all names are relative to `/var/spool/uucp`).
- The `STST` files have been replaced by `.Status/system`, and the files `ERRLOG` and `SYSLOG` have been replaced by `.Admin/errors` and `.Admin/xferstats`.
- Lock Files** Instead of having various lock files in `/var/spool/uucp`, a lock file for each device in use is created in the `/var/spool/locks` directory.
- Other Hidden Directories** In addition to the hidden directories mentioned above (`.Admin`, `.Log`, `.Status`), there is also a `.Corrupt` directory for work and execute files that are corrupted and cannot be processed, an `.Old` directory for old log files, a `.Sequence` directory for maintaining sequence numbers, a `.Workplace` directory used as a temporary workplace area, and a `.Xqtdir` directory for spooling remote executions.
- Upgrading your UUCP system** Before installing the new UUCP you should back up on tape all the old database files and spooled files and requests, as well as the old `/etc/passwd` file (if it contains entries for UUCP logins).
- After installing the new UUCP, bring the old files up from tape, and do the following (assuming that both old and new files reside in `/etc/uucp`— if this is not the case, modify the instructions below accordingly):

1. Copy the old `L.sys` file to the end of the new `Systems` file:

```
# cat L.sys >> Systems
```

2. Copy the old `L-dialers` file to the end of the new `Dialers` file using the method shown above.
3. Copy the old `L-devices` file to the end of the new `Devices` file using the method shown above.
4. Translate the information contained in the old files `L.cmds` and `USERFILE` to the end of the new `Permissions` file. Otherwise, for each entry in `/etc/passwd` that pertains to a UUCP site, generate an entry in `Permissions` of the form `LOGNAME=site_name`. This is the minimum that you need for other systems to be able to communicate with you. You may want to make their entries more elaborate in accordance with your needs.

Make sure you don't overwrite the new files. They contain invaluable information and examples. After you have done the above, change their ownership and permissions according to the following table:

Table 21-5 *Ownership and Permissions of the new files*

File	Owner	Group	Permission
Systems	uucp	uucp	400
Dialcodes	uucp	uucp	444
Dialers	uucp	uucp	444
Devices	uucp	uucp	444
Permissions	uucp	uucp	400

After doing the above, run the `/usr/lib/uucp/Cvt` script. This will try to take old `C.` and `D.` files in `/var/spool/uucp` and put them in the proper directories for the new UUCP. If you don't do this, they will be ignored.

Conversion Tips

Old `L-devices` entries should work properly in `Devices`, but all `ACU` lines should have the name of the modem (for example, `hayes` or `penril`) added to the end of the line, and all `DIR` lines should have `DIR` changed to `Direct`, and the word `direct` should be added to the end of the line.

Similarly, old `L.sys` entries should work properly in `Systems`, but for direct lines the first device specification should be `Direct`. Also, on time fields where there was a wait time specified, the `,` (comma) should be changed to `;` (semicolon). Notice that the time has a slightly different meaning than before. The retry algorithm is now an exponential backoff one with an initial time and a max retry time. If the `;time` is specified, that will always be the retry time. Otherwise, the backoff algorithm will be used.

Be aware that the `uucico` program no longer searches for a system to call. It must be called with the `-s system` option. A new program, `uusched`, does the search for work and calls remotes in random order by invoking `uucico` with the `-s` option.

21.22. 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 `/var/spool/uucp/.Admin/errors` file. STATUS errors are recorded in individual files for each machine found in the `/var/spool/uucp/.Status` directory.

UUCP ASSERT Error Messages

When a process is aborted, ASSERT error messages are recorded in `/var/spool/uucp/.Admin/errors`. These messages include the file name, sccsid, line number, and the text listed below. In most cases, these errors are the result of file system problems. The "errno" (when present) should be used to investigate the problem.

Table 21-1 *UUCP ASSERT Error Messages*

Error Message	Description/Action
CAN'T OPEN	An open() or fopen() failed.
CAN'T WRITE	A write(), fwrite(), fprintf(), etc. failed.
CAN'T READ	A read(), fgets(), etc. failed.
CAN'T CREATE	A creat() call failed.
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.
CAN'T CHMOD	A chmod() call failed.
CAN'T LINK	A link() call failed.
CAN'T CHDIR	A chdir() call failed.
CAN'T UNLINK	A unlink() call failed.
WRONG ROLE	This is an internal logic problem.
CAN'T MOVE TO CORRUPTDIR	An attempt to move some bad C. or X. files to the <code>/var/spool/uucp/.Corrupt</code> directory failed. The directory is probably missing or has wrong modes or owner.
CAN'T CLOSE	A close() or fclose() call failed.
FILE EXISTS	The creation of a C. or D. file is attempted, but the file exists. This occurs when there is a problem with the sequence file access. Usually indicates a software error.
No uucp server	A TCP/IP call is attempted, but there is no server for UUCP.
BAD UID	The user ID cannot be found in the <code>/etc/passwd</code> file. The file system is in trouble, or the <code>/etc/passwd</code> file is inconsistent.
BAD LOGIN_UID	Same as previous.

Table 21-1 *UUCP ASSERT Error Messages—Continued*

Error Message	Description/Action
BAD LINE	There is a bad line in the <code>Devices</code> 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 <code>gename.c</code> overflowed. A big/strange request was attempted. Contact your Sun representative.
TOO MANY SAVED C FILES	Same as previous.
RETURN FROM <code>fixline ioctl</code>	An <code>ioctl(2)</code> , which should never fail, failed. There is a system driver problem.
BAD SPEED	A bad line speed appears in the <code>Devices/Systems</code> files (<code>Class</code> field).
PERMISSIONS file: BAD OPTION	There is a bad line or option in the <code>Permissions</code> file. Fix it immediately!
PKCGET READ	The remote machine probably hung up. No action need be taken.
PKXSTART	The remote machine aborted in a non-recoverable way. This can generally be ignored.
TOO MANY LOCKS	There is an internal problem! Contact your Sun representative.
XMV ERROR	There is a problem with some file or directory. It is likely the spool directory, since the modes of the destinations were suppose to be checked before this process was attempted.
CAN'T FORK	An attempt to fork and <code>exec</code> failed. The current job should not be lost, but will be attempted later (<code>uuxqt</code>). No action need be taken.

UUCP STATUS Error Messages

Status error messages are messages that are stored in the `/var/spool/uucp/.Status` directory. This directory contains a separate file for each remote machine that your machine attempts to communicate with. These individual machine files contain status information on the attempted communication, whether it was successful or not. What follows is a list of the most common error messages that may appear in these files.

Table 21-2 UUCP STATUS Error Messages

Error Message	Description/Action
OK	Things are OK.
NO DEVICES AVAILABLE	There is currently no device available for the call. Check to see that there is a valid device in the <code>Devices</code> file for the particular system. Check the <code>Systems</code> 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 <code>Systems</code> file.
TALKING	Self explanatory.
LOGIN FAILED	The login for the given machine failed. It could be a wrong login/password, wrong number, a very slow machine, or failure in getting through the <code>Dialer-Token-Pairs</code> 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/machine name that does not agree with the <code>Permissions</code> 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 <code>/var/spool/uucp/.Admin/errors</code> file for the error message and refer to the section, <i>UUCP ASSERT Error Messages</i>
SYSTEM NOT IN <code>Systems</code> FILE	The system is not in the <code>Systems</code> file.
CAN'T ACCESS DEVICE	The device tried does not exist or the modes are wrong. Check the appropriate entries in the <code>Systems</code> and <code>Devices</code> files.
DEVICE FAILED	The open of the device failed.
WRONG MACHINE NAME	The called machine is reporting a different name than expected.
CALLBACK REQUIRED	The called machine requires that it calls your machine.

Table 21-7 *UUCP STATUS Error Messages—Continued*

Error Message	Description/Action
REMOTE HAS A LCK FILE FOR ME	The remote site has a LCK file for your machine. 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 machine, then the process that has a LCK file is hung.
REMOTE DOES NOT KNOW ME	The remote machine does not have the node name of your machine in its <code>Systems</code> file.
REMOTE REJECT AFTER LOGIN	The login used by your machine to login does not agree with what the remote machine was expecting.
REMOTE REJECT, UNKNOWN MESSAGE	The remote machine rejected the communication with your machine for an unknown reason. The remote machine may not be running a standard version of UUCP.
STARTUP FAILED	Login succeeded, but initial handshake failed.
CALLER SCRIPT FAILED	This is usually the same as "DIAL FAILED." However, if it occurs often, suspect the caller script in the <code>dialers</code> file. Use <code>Uutry</code> to check.

Part Four: Administrator's Reference

This part contains reference material that supplements the information given in Parts One, Two, and Three. Much of the text is taken from tutorials published by the University of California at Berkeley. Where applicable, references are cited.

Who Should Read This Part

Since this material is very technical in nature, it is recommended for experienced system administrators who want to tailor their systems to suit special needs, and those with programming experience who want to know more about the programs used to administer Sun computers.

What Is In This Part

Part Four contains reference material about the following subjects:

- Adding your own device drivers to the kernel configuration file.
- Adding IPC and TCP/IP parameters to the kernel configuration file.
- Analyzing crash dumps with the crash kernel debugger.
- Analyzing file system problems with the `fsck` utility.
- Customizing `sendmail` configuration files.
- Customizing terminals using `termcap`.

Part Four: Administrator's Reference — *Continued*

Advanced Administration Topics

This chapter covers a potpourri of subjects geared to advanced system and network administrators interested in customizing their configurations. Topics covered are:

- Adding device drivers that you have written into the kernel configuration file.
- Configuring multiple kernel images
- Configuring systems without source code
- Sharing object modules
- Building profiled kernels.
- Tuning IPC System Parameters
- Configurable TCP/IP Parameters
- Analyzing crash dumps and recovering from system crashes

The information in this chapter is supplemental to that provided in the chapter *Reconfiguring the System Kernel*. The text assumes that you are an advanced system administrator or programmer.

22.1. Files Used in Kernel Configuration

The configuration-build utility `/usr/etc/config` uses information from the following files to build the kernel:

<code>Makefile.src</code>	Generic makefile for the machine's model type.
<code>files.cmn</code>	List files common to many system architectures, required to build the basic kernel.
<code>files</code>	Machine-architecture specific files
<code>devices</code>	Name to major device mapping
<code>SYSTEM_NAME</code>	Describes characteristics of a specific system named <code>SYSTEM_NAME</code> . This is the kernel configuration file for the specific system, as created during reconfiguration process described in <i>Reconfiguring the System Kernel</i> .

From these files, `/usr/etc/config` generates the files needed to compile and link your kernel. One of these files is a *Makefile*, which you then use to actually

build the new system and install the kernel.

Adding Your Own Device Drivers

In addition to entries in the configuration file, new device drivers require entries in `/sys/sun/conf.c`, `/sys/sun[3,3x,4,4c]/conf/files`, possibly `/sys/sun/swapgeneric.c` and `/sys/sun[3,3x,4,4c]/conf/devices`. New devices also require you to add one or more new special files to `/dev`. Refer to the manual *Writing Device Drivers* for more information on these files.

To add a new device driver, you have to add lines in two of the three major sections of the GENERIC configuration file. These lines are options lines in the system identification section and device lines in the devices section, which are completely defined in *Reconfiguring the System Kernel*.

The options *optlist* Line

Device lines of this type should be located in the system identification section along with options lines provided by Sun. You use the options *optlist* argument when you want to add your own device drivers to the configuration file. This argument tells the kernel to compile the listed options (*optlist*) into the system. The items in *optlist* are separated by commas. A line of the form "options FUNNY, HAHA" yields `-DFUNNY -DHAHA` to the C compiler. An option may be given a value by following its name with "=", then the value enclosed in double quotes. None of the standard options use such a value.

In addition, options can be used to bring in additional files if the option is listed in the `/sys/sun[3,3x,4,4c]/conf/files`. All options should be listed in uppercase. In this case, no corresponding *option.h* will be created, as it would be using the corresponding *pseudo-device* method.

Adding Device Description Lines

The chapter *Reconfiguring the System Kernel* provides a general overview of device description lines. The sample kernel configuration files in that chapter illustrate how Sun uses these lines to configure standard equipment.

The syntax of the device description line is:

```
dev_type dev_name at connect_dev more_info
```

The arguments above are explained in detail, except the *more_info* argument, which is especially important when you add your own drivers. *more_info* is a sequence of the following:

- *csr addr*. This specifies the address of the csr (command and status registers) for a device. The csr addresses specified for the device are always the addresses within the bus type specified. For example, on the following line in a Sun-3 configuration file:

```
controller      si0 at vme24d16 ? csr 0x200000 priority 2 vector siintr 0x40
```

the address is 0x200000. The line indicates that the si0 controller exists on the

VME bus at address 0x200000.

When configuring your own drivers, the *csr* address must be specified for all controllers, and for all devices connected to a main system bus.

- *drive number*. This specifies which drive the line applies to. For example, the argument *sd0* refers to the first SCSI disk drive on a system's first SCSI controller, and the argument *xd4* refers to the first SMD disk drive on a system's second Xylogics 7053 controller.
- *flags number*. These are flags made available to the device driver, which are usually read during system initialization.
- *priority level*. This argument specifies the interrupt level of the interrupts that this device generates.
- *vector intr number [intr number...]*. This argument applies to devices that use vectored interrupts on VMEbus systems. *intr* specifies the vectored interrupt routine, and *number* specifies the corresponding vector to be used (64-255).

If you are adding non-standard devices to the configuration file, you can substitute a question mark (?) for a number in two places. You can put question marks on a *connect_dev* (for example, *xyz?*), or on a drive number (for example, *drive ?*). The system will automatically replace the ? with the appropriate value during boot up. This allows redundancy, since a single system can be built that will boot on different hardware configurations.

22.2. Configuring Systems Without Source

Object-only releases have binaries for standard system modules in the directory `/usr/sys/sun[3,3x,4,4c]/OBJ`. Using these binaries you can create new configurations and add new device drivers to the kernel.

You can make any changes you wish to the configuration file, provided you do not configure in more devices of a particular type than are allowed by the distributed object code in `/usr/sys/sun[3,3x,4,4c]/OBJ`. Attempting to do so will not be detected and may cause the kernel to appear to work but have only occasional failures. Double check the `.h` files in `/usr/sys/sun[3,3x,4,4c]/OBJ` if you change the number of devices configured for any standard drivers.

22.3. Sharing Object Modules

If you have many kernels that are all built on a single machine, there are at least two approaches to save time in building kernel images. The best way is to have a single kernel image being run on all machines. This is attractive because it minimizes disk space used and time required to rebuild kernels after making changes. However, often one or more systems requires a separately configured kernel image. This may be due to limited memory (building a kernel with many unused device drivers wastes core), or to configuration requirements (one machine may be a development machine where accounting is not needed, while another is a production machine where it is). In these cases kernels could share relocatable object modules that are not configuration dependent. Most of the modules in the directory `/usr/sys/os` are of this type.

Note: You can only build sharable object modules if your company has licensed the SunOS source code distribution from Sun.

To share object modules, first build a GENERIC kernel. Then, for each kernel, configure as before, but before recompiling and linking, type `make links` specifically, after the `make depend` command and before the `make` that creates your kernel, as explained in *Reconfiguring the System Kernel*.) This causes the kernel source to be searched for source modules that can be shared between kernels and generates symbolic links in the current directory to the appropriate object modules in the directory `../GENERIC`. A shell script, `make-links` is generated and executed with this request and may be checked for correctness.

The file `/sys/conf/common/defines` contains a list of symbols that are relatively safe to ignore when checking the source code for modules that may be shared. Note that this list includes the definitions used to conditionally compile in the virtual memory tracing facilities, and the trace point support used only rarely. It may be necessary to modify this list to reflect local needs. Also, as described previously, interdependencies that are not directly visible in the source code are not caught. Thus if you place per-system dependencies in an include file, they will not be recognized.

Building Profiled Kernels

Note: You can only build profiled kernels if your company has licensed the SunOS source code distribution from Sun.

It is simple to configure a kernel that will automatically collect profiling information as it operates. The profiling data can be collected with `kgmon(8)` and processed with `gprof(1)` to obtain information about how the kernel operates. Profiled kernels maintain histograms of the program counter as well as the number of invocations of each routine. The `gprof(1)` command generates a dynamic call graph of the executing kernel and propagates time spent in each routine along the arcs of the call graph.

To configure a profiled kernel, use the `-p` option with the `config` program. A profiled kernel is about 5-10 percent larger in its text space due to the calls to count the subroutine invocations. When the kernel executes, the profiling data is stored in a buffer that is 1.2 times the size of the text space. The overhead for running a profiled kernel varies. Under normal load, this may be anywhere from 5-25 percent of the kernel time spent in the profiling code.

Note that kernels configured for profiling should not be shared as described above unless all the other shared kernels are also to be profiled.

22.4. Rules for Defaulting System Devices

This section covers the rules the `/usr/etc/config` program uses to define underspecified locations of system devices.

When the `/usr/etc/config` program processes a `config` line that does not fully specify the location of the root file system, swap or paging area, and device for system dumps, it applies a set of rules to define those values left unspecified. The following list of rules is used:

1. If a root device is not specified, the swap specification must indicate a generic system is to be built.
2. If the root device does not specify a unit number, it defaults to unit 0.

3. If the root device does not include a partition specification, it defaults to the a partition.
4. If no swap area is specified, it defaults to the b partition of the root device.
5. If no device is chosen for system dumps, the swap partition is selected (see below to find out where dumps are placed within the partition).

The following table summarizes the default partitions selected when a device specification is incomplete, for example, `xy0`.

Type	Partition
root	a
swap	b
dump	b

You can use `swapon(8)` to add additional swapping devices/files while your system is in use. The kernel only needs to know the primary swap device at boot time. Use `swapon` to add secondary devices as needed.

Finally, system dumps are automatically taken after a system crash, provided the device driver for the dumps device supports this. The dump contains the contents of memory, but not the swap areas. Normally the dump device is a disk, in which case the information is copied to a location near the back of the partition. The dump is placed in the back of the partition because the primary swap and dump device are commonly the same device and this allows the system to be rebooted without immediately overwriting the saved information. When a dump has occurred, the system variable `dumpsize` is set to a non-zero value indicating the size (in bytes) of the dump. The `savecore(8)` program then copies the information from the dump partition to a file in a “crash” directory and also makes a copy of the system that was running at the time of the crash (usually `/vmunix`).

Configuration File Grammar

This section covers the grammar used by the `/usr/etc/config` program to parse the input kernel configuration file.

The following grammar is a compressed form of the actual `yacc(1)` grammar used by the `/usr/etc/config` program to parse configuration files. Terminal symbols are shown all in uppercase, literals are emboldened; optional clauses are enclosed in brackets, `[` and `]`; zero or more instances are denoted with `*`. `/* lambda */` and `/* epsilon */` denote the empty string.

```

Configuration ::= [ Spec ; ]*

Spec ::= Config_spec
      | Device_spec
      | trace
      | /* lambda */

/* configuration specifications */

Config_spec ::= machine ID
              | cpu ID
              | options Opt_list
              | ident ID
              | System_spec
              | maxusers NUMBER

/* system configuration specifications */

System_spec ::= config ID System_parameter [ System_parameter ]*

System_parameter ::= swap_spec | root_spec | dump_spec

swap_spec ::= swap [ on ] swap_dev

swap_dev ::= dev_spec [ size NUMBER ]

root_spec ::= root [ on ] dev_spec

dump_spec ::= dumps [ on ] dev_spec

dev_spec ::= dev_name | major_minor

major_minor ::= major NUMBER minor NUMBER

dev_name ::= ID [ NUMBER [ ID ] ]

/* option specifications */

Opt_list ::= Option [ , Option ]*

Option ::= ID [ = Opt_value ]

Opt_value ::= ID | NUMBER

/* device specifications */

Device_spec ::= device Dev_name Dev_info Int_spec
              | disk Dev_name Dev_info
              | tape Dev_name Dev_info
              | controller Dev_name Dev_info [ Int_spec ]
              | pseudo-device Dev [ NUMBER ]

```

```

Dev_name ::= Dev NUMBER

Dev_info ::= Con_info [ Info ]*

Con_info ::= at Dev NUMBER
           | at nexus NUMBER

Info ::= csr NUMBER
       | drive NUMBER
       | slave NUMBER
       | flags NUMBER

Int_spec ::= priority NUMBER
          | priority NUMBER vector ID NUMBER [ ID NUMBER ] *
          | /* epsilon */

```

Lexical Conventions

The terminal symbols are loosely defined as:

ID

One or more alphabets, either upper or lower case, and underscore, `_`.

NUMBER

Approximately the C language specification for an integer number. That is, a leading `0x` indicates a hexadecimal value, a leading `0` indicates an octal value, otherwise the number is expected to be a decimal value. Hexadecimal numbers may use either upper or lower case alphabets.

In special instances a question mark (`?`) can be substituted for a `NUMBER` token. This is used to effect wildcarding in device interconnection specifications.

Comments in configuration files are indicated by a `#` character at the beginning of the line; the remainder of the line is discarded.

A specification is interpreted as a continuation of the previous line if the first character of the line is `tab`.

Data Structure Sizing Rules

This section explains the rules for sizing kernel data structures, both those calculated at compile time and those calculated at boot time.

Certain kernel data structures are sized at compile time according to the maximum number of simultaneous users expected, while others are calculated at boot time based on the physical resources present, such as memory. This section lists both sets of rules and also includes some hints on changing built-in limitations on certain data structures.

Compile Time Rules

The file `/usr/sys/conf.common/param.c` contains the definitions of almost all data structures sized at compile time. This file is copied into the directory of each configured kernel to allow configuration-dependent rules and values to be maintained. The rules implied by its contents are summarized below (Here `MAXUSERS` refers to the value defined in the configuration file in the `maxusers` rule.)

nproc	The maximum number of processes that may be running at any time. It is defined to be $10 + 16 * \text{MAXUSERS}$ and referred to in other calculations as <code>NPROC</code> .
ninode	The maximum number of UFS files in the file system that may be active at any time. This includes files being used by users, as well as directory files being read or written by the system and files associated with bound sockets in the IPC domain. This is defined as $(\text{NPROC} + 16 + \text{MAXUSERS}) + 64 + 12 * \text{MAXUSERS}$.
nfile	The number of file table structures. One file table structure is used for each open, unshared, file descriptor. Multiple file descriptors may refer to a single file table entry when they are created through a <code>dup</code> call, or as the result of a <code>fork</code> . This is defined to be $16 * (\text{NPROC} + 16 + \text{MAXUSERS}) / 10 + 64 \text{ if } \text{NWIN} = 0,$ $16 * (\text{NPROC} + 16 + \text{MAXUSERS}) / 5 + 64 \text{ if } \text{NWIN} > 0,$
ncallout	The number of callout structures. One callout structure is used per internal system event handled with a timeout. Timeouts are used for terminal delays, watchdog routines in device drivers, protocol timeout processing, and so on. This is defined as $16 + \text{NPROC}$.
nclist	The number of c-list structures. C-list structures are used in terminal I/O. This is defined as $100 + 16 * \text{MAXUSERS}$.
ndquot	The number of <code>dquot</code> structures allocated. <code>dquot</code> structures are present only when disk quotas are configured in the system. One <code>dquot</code> structure is required per user, per active file system quota. That is, when a user manipulates a file on a file system on which quotas are enabled, the information regarding the user's quotas on that file system must be in core. This information is cached, so that not all information must be present in core all the time. This is defined as $(\text{MAXUSERS} * \text{NMount}) / 4 + \text{NPROC}$, where <code>NMount</code> is the maximum number of mountable file systems.

22.5. Tuning IPC System Parameters

The System V Inter-Process Communications (IPC) extensions to UNIX are now implemented in the SunOS operating system. These IPC extensions provide mechanisms for message passing, semaphores, and shared memory.

Data structures that describe and control various IPC functions are allocated in the SunOS kernel at system initialization and remain resident in memory as long as the operating system is running. The size of these structures is determined by a variety of tunable parameters in the system configuration file. Some IPC parameters also exist to control and limit the resources dynamically allocated by IPC

subsystems.

This section describes these parameters and gives some guidelines for tuning them. Before attempting to modify any of these parameters, you should be thoroughly familiar with the capabilities provided by the IPC system, and with your particular application's needs.

Refer to the *AT&T System V Interface Definition (SVID)* and Sun's *SunOS Reference Manual* for a detailed description of the IPC system interface. But note that there are now two separate IPC subsystems – the one from 4.x BSD involving sockets, and the one from System V discussed here.

Why Reconfigure IPC Parameters?

The SunOS operating system, as shipped, is configured to support a modest level of IPC activity. Applications that require more IPC resources than are provided in the distributed system must be run with reconfigured kernels. In general, Sun recommends that you reconfigure all system kernels as described earlier in *Reconfiguring the System Kernel*, in order to reduce the base-level memory requirements. The following paragraphs assume a working knowledge of the system configuration procedure.

IPC parameters may be tuned by editing the system configuration file `/usr/sys/sun[3,3x,4,4c]/conf/KERNEL_NAME` and rebuilding the kernel. In order to adjust the default setting of a particular parameter, insert a line of the following form into the configuration file:

```
options OPTION_NAME = OPTION_VALUE
```

Default values are assumed for all unspecified parameters.

Note: Do not change the `options` lines that refer to the IPC subsystems unless you wish to disable the entire subsystem.

Users of System V may note that there is not an exact correspondence between Sun's tunable parameters and those found in the AT&T System V release. This is because several implementation algorithms have been changed, rendering some of the parameters meaningless. In particular, certain static structures in System V are allocated dynamically by the Sun kernel, obviating the need for configurable limits. In future Sun releases, there are likely to be even fewer tunable parameters. It should be noted that although some parameters have been omitted, there are no semantic differences in the IPC system call operation.

For the remaining tunable parameters, reasonable values may be estimated using information about the application programs' IPC usage. The following sections specify the default values and attempt to give tuning guidelines. It is generally a good idea to write application programs that recover gracefully from resource allocation failures, so that system configuration requirements surface early in the development cycle. Applications that require large amounts of IPC resources are often poorly designed and should be carefully reviewed before making drastic increases in the IPC parameters.²

² The total amount of memory available for user processes may be estimated using the `vmstat(8)` program.

IPC Message Parameters

These parameters control system characteristics associated with inter-process message passing.

MSGPOOL

MSGPOOL defines the size (in kilobytes) of the kernel message memory pool. All queued IPC messages are stored in this pool. The behavior of the *msgsnd(2)* system call when the message pool is full depends on the value of the *msgflg* argument; see *msgop(2)*. Attempts to queue messages larger than the message pool return *EINVAL*.

The IPC message pool is allocated at system initialization and is never made available for other uses. Thus, tuning the **MSGPOOL** parameter involves a tradeoff between the performance of processes that depend upon a high level of message activity, and the degradation of overall system performance due to wasted memory. This parameter may not exceed 255.

```
options      MSGPOOL=8
```

MSGMNB

MSGMNB defines the maximum number of message bytes that may be queued on any particular message queue. Attempts to queue messages that would exceed this limit either sleep or return *EAGAIN*; see *msgop(2)*. **MSGMNB** is used as a default value for *msg_qbytes* when message queues are created; this limit may be lowered by any process but only the superuser may raise the limits on a particular message queue; see the *msgctl(2)* option named *IPC_SET*.

```
options      MSGMNB=2048
```

MSGMNI

MSGMNI defines the maximum number of uniquely identifiable message queues that may exist simultaneously. Attempts to create more than **MSGMNI** message queues return *ENOSPC*; see *msgget(2)*. Each increment of **MSGMNI** reserves 48 bytes of kernel memory.

```
options      MSGMNI=50
```

MSGTQL

MSGTQL defines the total number of undelivered messages that may exist at any instant. Attempts to queue more than **MSGTQL** messages either sleep or return *EAGAIN*; see *msgop(2)*. Since zero-length messages are allowed, this limit could theoretically be set arbitrarily high. Each increment of **MSGTQL** reserves 12 bytes.

```
options      MSGTQL=50
```

IPC Semaphore Parameters

These parameters control system characteristics associated with inter-process semaphores.

SEMMNI

SEMMNI defines the maximum number of uniquely identifiable semaphore clusters that may exist simultaneously. Attempts to create more than **SEMMNI** semaphore clusters return *ENOSPC*; see *semget(2)*. Although **SEMMNI** may be set arbitrarily

high, there is no reason to set it to be larger than `SEMMNS`. Each increment of `SEMMNI` reserves 32 bytes of kernel memory.

```
options      SEMMNI=10
```

SEMMNS

`SEMMNS` defines the maximum number of semaphores in the system. Attempts to create semaphore clusters when there are not enough semaphores available result in an `ENOSPC` error; see *semget(2)*. Attempts to create semaphore clusters with more than `SEMMNS` semaphores return `EINVAL`. Each increment of `SEMMNS` reserves 8 bytes.

```
options      SEMMNS=60
```

SEMUME

`SEMUME` defines the maximum number of semaphores (per process) that may simultaneously have non-zero adjust-on-exit values. The adjust-on-exit values are manipulated when semaphore operations are requested in conjunction with the `SEM_UNDO` flag. Attempts to exceed this limit return `EINVAL`; see *semop(2)*. The value of `SEMUME` affects the number of bytes allocated for semaphore undo structures (see `SEMMNU` below). The value of `SEMUME` must be less than 32768.

```
options      SEMUME=10
```

SEMMNU

`SEMMNU` defines the maximum number of processes that may simultaneously be using the IPC `SEM_UNDO` feature. Attempts to exceed this limit result in an `ENOSPC` error; see *semop(2)*. There is no reason to set `SEMMNU` larger than the maximum number of processes that can run on the system (approximately $16 * \text{maxusers}$, where `maxusers` is configurable, defaulting to 8). Therefore, `SEMMNU` need not exceed 128, under normal circumstances. Each increment of `SEMMNU` allocates $14 + (8 * \text{SEMUME})$ bytes.

```
options      SEMMNU=30
```

IPC Shared Memory Parameters

These parameters control system characteristics associated with inter-process shared memory.

SHMMNI

`SHMMNI` defines the maximum number of shared memory segments that may simultaneously exist in the system. Attempts to exceed this limit return `ENOSPC`; see *shmget(2)*. Each increment of `SHMMNI` reserves 42 bytes.

```
options      SHMMNI=100
```

SHMSIZE

`SHMSIZE` defines the size, in kilobytes, of the largest single shared memory segment. Attempts to exceed this limit return `EINVAL`; see *shmget(2)*. Although indiscriminate allocation of shared memory can exhaust system resources (swap space for instance), there is no enforced limit on the total amount of shared memory that may be allocated. However, in order to protect against simple program errors, this parameter limits the size of individual shared segments. The default value, equivalent to 1 MByte, should be sufficient for most applications.

```
options    SHMSIZE=1024
```

Named Pipe Parameters

These parameters control system characteristics associated with FIFO special files (named pipes).

FIFOCNT

FIFOCNT determines the maximum number of FIFOs that may be in use in the system at any given time. Attempts to write to more than FIFOCNT simultaneous FIFOs will block until some FIFOs are closed, releasing system resources. In the current implementation, FIFO buffers are allocated in non-paging memory, and can lead to system lockup if indiscriminately allocated. This restriction may be removed in a future release.

```
options    FIFOCNT=10
```

22.6. TCP/IP Configuration Options for SunOS 4.1

In Release 4.1, you can change parameters in the TCP/IP software. Table 22-1 lists these parameters, which are stored in the file `netinet/in_proto.c` in the release kernel build directory (usually `/usr/sys`). The first column of Table 22-1 gives the variable name, and second column its default value. The third column, if given, is the option that can be set in the kernel config file. To use a config option, add the line:

```
options OPTION=value
```

to your kernel config file and run `config` and make to rebuild the kernel. If no config option is given, then you need to make a copy of `in_proto.c`, and then edit `in_proto.c` to change the variable to the given value, and rebuild the kernel. Experienced administrators may also be able to use `adb` to patch these variables in already built or even running kernels, but that is not recommended.

For example, adding the line:

```
options IPFORWARDING=1
```

will force IP forwarding to be on. Editing the line of `in_proto.c` to read as follows:

```
int udp_cksum = 1;
```

will cause UDP to generate and check UDP checksums.

Table 22-1 TCP/IP Default Parameters

<i>C Language Variable</i>	<i>Value</i>	<i>config Options Line</i>
ip_forwarding	0	IPFORWARDING
ip_subnetslocal	1	SUBNETSARELOCAL
ip_sendredirects	1	IPSENDREDIRECTS
ip_dirbroadcast	1	DIRECTED_BROADCAST
ip_printfs	0	
tcp_default_mss	512	
tcp_sendspace	4096	
tcp_recvspace	4096	
tcp_keeplen	1	
tcp_ttl	60	
tcp_nodelack	0	
tcp_keepidle	14400	
tcp_keepintvl	150	
udp_cksum	0	
udp_ttl	30	
udp_sendspace	9000	
udp_recvspace	18032	

Descriptions of TCP/IP Parameters

ip_forwarding	Controls whether or not to forward IP packets that this machine thinks are intended for another machine. This parameter has three possible states: the default is 0, which will automatically forward only if the machine has more than one interface given an IP address. The value 1 causes forwarding to always be enabled, and -1 causes forwarding to never be enabled. Use the value -1 for machines that are used as relays (application-level gateways) for security purposes.
ip_subnetslocal	Determines if destinations on other subnets of the same IP network as this host are to be considered “local”. This is on by default. Change to zero to consider them “non-local”. This parameter is used to determine if the TCP Maximum Segment Size should match the network maximum transfer unit (MTU) size (for higher performance) or if the MSS should be selected conservatively (as the value in the tcp_default_mss) to avoid fragmentation problems. This is also used to determine if HOST or NET redirect ICMP messages should be sent, and if the EHOSTUNREACH or ENETUNREACH error should be returned on non-routable IP packets.
ip_sendredirects	Determines if ICMP redirect messages are to be sent if forwarding sends a packet out the same interface over which it comes. Normally this should be enabled (1).

<code>ip_dirbroadcast</code>	Allows directed broadcast. By default, packets can be routed from network to another, even if they result in a broadcast on the destination network. Set this parameter to zero to prevent directed broadcasting.
<code>ip_printfs</code>	Print out when packets are forwarded or ICMP redirects are sent. Used only for debugging.
<code>tcp_default_mss</code>	The default Maximum Segment Size (MSS) for TCP, in bytes. If the destination is on the same network, the MSS is based on the Maximum Transmission Unit (MTU) size of the network. This value is used in all other cases. The conservative value of 512 avoids fragmentation problems. For improved performance through routers to other high-speed networks (between Ethernets or FDDI, for example) set to 1024, 2048, 4096, for example.
<code>tcp_sendspace</code>	Number of bytes that the user can send to a TCP socket buffer before being blocked. The default value of 4096 can be changed on a given socket with the <code>SO_SNDBUF</code> ioctl.
<code>tcp_recvspace</code>	Number of bytes that the remote TCP can send into a socket buffer before being blocked. This determines the TCP window size presented to the other side. The default value of 4096 can be changed on a given socket with the <code>SO_RCVBUF</code> ioctl.
<code>tcp_keepen</code>	Size of the TCP "keep alive" packets. By default it is set to one for compatibility with older releases, although zero probably makes it closer to the TCP specifications.
<code>tcp_ttl</code>	The "Time To Live" field of TCP segments in seconds. The default is 60 seconds.
<code>tcp_nodelack</code>	TCP normally tries to conserve network bandwidth at the expense of response time, by avoiding acknowledgement of small packets. Setting this parameter to one defeats this mechanism.
<code>tcp_keeidle</code>	The time in half-seconds after a connection becomes idle when "keep-alive" probes start being sent. The default of two hours should be high enough to reduce the amount of unnecessary network traffic. Reduce to produce quicker reset when remote hosts become unreachable.
<code>tcp_keepintvl</code>	The interval in half-seconds at which keep-alive probes are sent. The default is a minute and fifteen seconds.
<code>udp_cksum</code>	Enables the checking and generation of UDP checksums. Normally this parameter is turned off to increase NFS performance. Set it to one when using NFS over long-haul networks. Note that TCP and IP checksums are not

	optional.
<code>udp_ttl</code>	The default “Time To Live” in seconds for UDP datagrams.
<code>udp_sendspace</code>	Number of bytes that the user can send to a UDP socket buffer at one time. The default value of 9000 can be changed on a given socket with the <code>SO_SNDBUF</code> ioctl.
<code>udp_recvspace</code>	Number of bytes that the remote UDP can send into a socket buffer before being dropped. The default value of 18032 can be changed on a given socket with the <code>SO_RCVBUF</code> ioctl.

22.7. The Crash Kernel Debugger

The SunOS Release 4.1 contains a utility to examine the memory images of a live or crashed system kernel. This tool can be used to examine the control structures, active tables, and other pertinent information regarding the operation of a kernel.

The `crash` utility is found in the `/etc` directory. By using `crash`, you can examine the system kernel image of either a crashed or live kernel, much as you would use `dbx` to examine the binary code of a program. Because `crash` is a debugging tool, its usefulness to a system administrator is mostly in determining where a system has problems. To use `crash` to its full potential requires a rather in depth knowledge of the kernel and how it operates. Since this manual is not meant to be a discussion of the internals of the kernel, only those few aspects of `crash` which are of immediate aid to a system administrator are shown here. For further details on the operation of the `crash` utility, please refer to the man page `crash(8)`.

Preserving Core Images

If a system crashes for any reason, the first clue that a system administrator has to what is the problem is the error message that is written to the console. Usually, this will be something like:

```
panic: a short description of the error
```

Common system panic messages, and their causes

- `io error in pushhard io error in swap`
The system encountered an error trying to write to the paging device or trying to read critical information from the disk drive. This indicates that the drive is failing or failed.
- `timeout table overflow`
The system ran out of entries in the timeout table. To fix, simply increase the value of `ncallout` in the file `param.c` in the kernel source directory `/usr/kvm/sys/conf.common`.
- `init died`
The process which admits new users to the system has died for some reason. The fix is a reboot, which is done automatically.

- `trap type type, pid process_id, pc = program counter, sr = status register, context context-number`

The internal consistency checks of the kernel have detected a problem. The nature of the error is shown by the *type* of trap made, while the remaining data (pid, pc, sr, and context) have to do with where the problem is. WRITE THESE NUMBERS DOWN. This is the info needed when you examine the core image using `crash`.

This may be all you need to determine the problem. Frequently, however, this message is only one of the clues you need to find, and eventually fix, the problem.

The System Core Image

The best source of information about what is wrong with a kernel is the *core image* of kernel. This is a “snapshot” of the kernel containing all of the tables, lists, and control structures that make up the heart of the SunOS operating system. When a system crash occurs, this information — normally held in the pseudo-device `/dev/mem`— is written to the tail end of the primary swap area of memory. Since the swap area is actually held on disk, this preserves the core image as the system dies.

To retrieve this core image, the program `savecore` must be run. `savecore` is designed to run from the `rc.local` script. Since `savecore` is not normally run, you need to uncomment the following sections of `rc.local`:

```
# Default is to not do a savecore
#
#mkdir /var/crash/`hostname`
#           echo -n 'checking for crash dump... '
#intr savecore /var/crash/`hostname`
#           echo ''
```

When you have finished doing that, the pertinent section of `rc.local` should look like the following:

```
# Default is to not do a savecore
#
mkdir /var/crash/`hostname`
           echo -n 'checking for crash dump... '
intr savecore /var/crash/`hostname`
           echo ''
```

Now, when the system reboots, any core images will be saved to the directory `/var/crash/hostname`, where *hostname* is the name of the machine which has crashed. The actual file saved is called `vmcore.n`, where *n* is a number which increments whenever a core image is saved. Therefore, if the host named `einstein` crashes, and the `rc.local` script has been modified to include running `savecore`, then the following occurs:

- If there is no directory `/var/crash/einstein`, then that directory is created.

- The core image is saved in that directory, using the filename `vmcore.1`. If one or more core files already exists, then the file will be `vmcore.n+1`, where *n* is the highest numbered core in the directory.

What Happens When a File System Is Nearly Full?

If a file system is very full when `savecore` attempts to salvage a core image from swap, then that core image may not be saved. To determine the amount of available space left, `savecore` reads a file from the `/var/crash/`hostname`` directory called `minfree`. This file contains the minimum amount of free space that is necessary to save that core image. If the number in `/var/crash/`hostname`/minfree` is less than the available space on the file system which contains the directory `/var/crash/`hostname``, then the core image is not saved.

Starting `crash`

The `crash` utility is started using the following syntax:

```
/etc/crash [ -d dumpfile ] [ -n namelist ] [ -w outputfile ]
```

The three optional arguments are:

- d The name of the core image. If no filename is given, the current core is used, as read from `/dev/mem`.
- n The file containing the symbol table for the core image. The default is `/vmunix`.
- w The file name to which the output from `crash` will be written. The default is the standard output.

When `crash` starts up, the values for the three options will be printed. If any of the arguments are omitted, then the default for that argument is taken. Finally, a prompt “>” will be given.

Useful `crash` Commands

`crash` has over 50 separate commands, so only a few will be discussed here. To see all of the active `crash` commands, use the built in help feature, the “?” command. With all of the commands to `crash`, you can give them the “-w” argument to redirect the output to a file. This redirection will only apply to that command, however. Also, any command preceded by an exclamation point (i.e. “!”) will sent to the shell instead of being executed by `crash`.

the `base` command

This takes the argument, which is a value in some base (as indicated with a preceding `0x` [hex], `0` [octal], `0b` [binary], or nothing for decimal) and prints that value out in the all of the appropriate bases.

Note: All values given to `crash` accept the preceding base indicators as explained in the `base` command.

the `defproc` command

This allows you to see what the current process being examined is, plus the ability to specify which of the current processes you wish to examine. To change the current process, use the slot number of the process, as determined by the `proc` command.

the mount command

Shows the current mounted disks. This is useful when you are experiencing disk problems and you believe the disk driver may be at fault.

the nm command

Used to examine the value of a symbol as found in the symbol table. Again, really only useful if you are working with a section of kernel or driver code you are familiar with.

the file command

This shows all the open files, and what kind of operations are being conducted on those files. If a kernel has crashed due to a privilege violation or some other file operation, this can be used to see what might have been wrong.

the help command

You can get a short explanation of any of the active commands using the command `help`, followed by the command you wish help with.

the pcb command

When a crash has occurred due to a bus error or the like, the `pcb` command can be used to show the process control block for that process. The output of `pcb` is all of the registers in the CPU, the process status register, and a pointer to the bus error location.

the proc command

This prints the table of all active processes at the time of the crash (for a core image), or at the current moment (for a live kernel). And individual entry in the process table can be examined by various arguments. A process can be found by PID, by the number of its entry in the process table, or all runnable processes can be shown with the “-r” option. This command is useful when an offending process is suspected of causing the crash.

the quit command

Terminates the crash session.

the redirect command

Allows a different output file to be selected, rather than the current one. This performs the same as a “-w” option, but remains in effect for all later commands.

the size command

Shows the size of the structure specified by the argument. If there is no argument, then all structures which have sizes associated with them are listed. This is used when you suspect that the crash was caused by a structure filling up. If the size of the structure is too small, then the appropriate structure will have to be made bigger in the kernel code.

the stack command

This dumps the stack of either the user, the kernel, or a process, as specified by the “-u”, “-k”, or the “-p” arguments, respectively. When a stack overflows and causes a panic, this command can be used to determine if a single process has filled the stack to an abnormal degree.

the `stat` command

Shows the relevant status of the kernel being examined. Usually used first when preparing a listing of a crash session (using the “-w” option, or the `redirect` command) to identify that listing.

the `trace` command

Lists the last functions that the kernel was performing before the crash. If the crash did not produce a “panic” message with the information regarding where the kernel died, this will be of aid.

What do I do now?

If the problem has been determined, you will need to fix it or your machine will undoubtedly crash again. For some problems, such as a bad drive, the fix is self-evident: fix or replace the drive. For stack overflows, driver malfunctions, or other problems, you must fix the problem in the relevant section of code, and then rebuild your kernel. These topics are covered at the beginning of this chapter and in *Reconfiguring the System Kernel*. *Writing Device Drivers*

Customizing `sendmail` Configuration Files

`sendmail` implements a general internetwork mail transport facility, featuring aliasing and forwarding, automatic routing to network gateways, and flexible configuration. To do this, it uses a configuration file. Sun supplies standard configuration files that most sites can use. The chapter on *Administering Electronic Mail* explains how to set up an electronic mail system based on these standard files. However, you might have to tailor the configuration files to fit your site's needs. This chapter explains how to customize the `sendmail` configuration files.

`sendmail` can use different types of communications protocols, such as TCP/IP and `uucp`. It also implements an SMTP server, message queueing, and mailing lists. Name interpretation is controlled by a pattern-matching system that can handle both domain-based naming and *ad hoc* conventions.

Sections in this chapter discuss the following subjects:

- How to do a basic `sendmail` installation.
- Day-to-day information you need to know to maintain your mail system. If you have a relatively typical site, these two topics contain sufficient information for you to install `sendmail` and keep it running smoothly.
- Command line arguments to `sendmail`.
- `sendmail` parameters that you can alter.
- In-depth information on the configuration file. This section provides information for those sites that need to write their own configuration file.
- Brief but detailed explanation of several lesser-used features of `sendmail`.

For information on routers, gateways, and setting up an internetwork, refer to the chapter on *The SunOS Network Environment*.

The domain technique separates the issue of physical versus logical naming. Refer to the chapter on *Administering Domain Name Service*, for a complete description of Internet domain naming conventions. Additionally, `sendmail` can accept old arbitrary name syntaxes, resolving ambiguities using heuristics you specify, as well as domain-based naming. It helps to convert messages between disparate naming schemes.

Certain special cases can be handled by *ad hoc* techniques, such as providing network names that appear local to hosts on other networks. For example, "user@host" is left to right syntax, and "host!host" is right to left syntax.

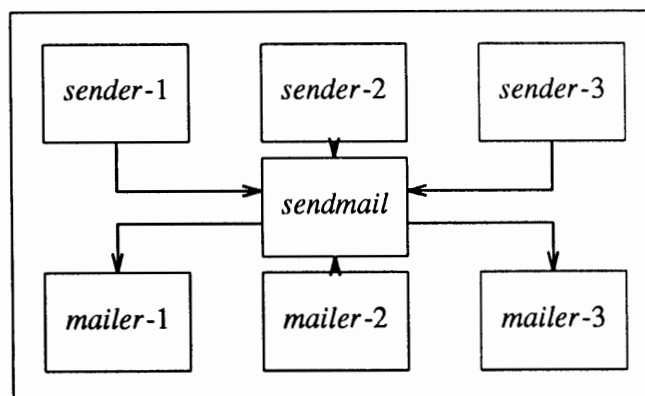
23.1. sendmail Features

sendmail was designed to implement the following features:

1. It implements System V mail, Bell version 7 mail, and Internet mail.
2. sendmail is reliable. It was designed to correctly deliver every message so that it can be disposed of. This way no message should ever be completely lost.
3. sendmail uses existing software to do actual delivery whenever possible.
4. sendmail provides easy expansion to fairly complex environments, including multiple connections to a single network type (such as with UUCP or Ethernets). This goal requires consideration of the contents of a name as well as its syntax to determine which mailer to use.
5. Mail configuration does not need to be compiled into the code. A single compiled program should work at most sites. Besides the simple problems that occur when any program gets recompiled in a different environment, many sites would probably alter anything that they will be recompiling anyway.
6. sendmail lets various groups maintain their own mailing lists. Additionally, it allows individuals to specify their own forwarding without modifying the domain-wide alias file (typically located in the domain-wide aliases maintained by yp).
7. sendmail enables each user to specify a custom mailer to process incoming mail. This feature allows functions such as returning an "I am on vacation" message. Refer to `vacation(1)` in the *SunOS Reference Manual*
8. sendmail minimizes network traffic by batching recipients to a single host where possible, without assistance from the user.

Figure 23-1 shows how the sendmail system is organized.

Figure 23-1 Sendmail System Structure



The user interacts with a mail generating and sending program. When the mail is submitted, the generator calls `sendmail`, which routes the message to the correct mailer(s). Since some of the senders may be network servers and some of the mailers may be network clients, `sendmail` may be used as an Internet mail gateway.

23.2. `sendmail` Overview

`sendmail` neither interfaces with the user nor does actual mail delivery. Rather, it collects a message from a program such as `Mail` or `MH` edits the message as required by the destination mailer, and calls appropriate mailers to do delivery or queuing for network transmission³. This discipline allows the insertion of new mailers at minimum cost. In this sense `sendmail` resembles the Message Processing Module (MPM) of [Postel79b].

Interfaces to the Outside World

There are three ways `sendmail` can communicate with the outside world, both in receiving and in sending mail. These are using the conventional UNIX argument vector/return status, speaking over a pair of UNIX pipes, and speaking SMTP over a TCP connection.

Argument Vector/Exit Status

The argument vector (command name and arguments) is the standard UNIX method of communicating with a process. A list of recipients is sent in the argument vector, and the message body is sent on the standard input. Anything that the mailer prints is simply collected and sent back to the sender if there were any problems. The exit status from the mailer is collected after the message is sent, and a diagnostic is printed if appropriate.

SMTP over Pipes

The SMTP protocol can be used to run an interactive lock-step interface with the mailer. A subprocess is still created, but no recipient names are passed to the mailer via the argument list. Instead, they are passed one at a time in commands sent to the processes' standard input. Anything appearing on the standard output must be a standard SMTP reply code.

SMTP over a TCP Connection

This technique is similar to the previous technique, except that it uses a TCP connection. Refer to Section 4 of the *SunOS Reference Manual*. This method is exceptionally flexible in that the mailer need not reside on the same machine. It is normally used to connect to a `sendmail` process on another machine.

How `sendmail` Works

When a sender wants to send a message, it issues a request to `sendmail` using one of the three methods described above. `sendmail` operates as described in the steps below.

Argument Processing and Address Parsing

If `sendmail` is called using one of the first two techniques above, the arguments are first scanned and option specifications are processed. Recipient names are then collected, either from the command line or from the SMTP command, and a list of recipients is created. Aliases are expanded at this step, including mailing lists. As much validation as possible of the remote recipient is done at this step: syntax is checked, and local recipients are verified, but detailed checking of host

³ except when mailing to a file, when `sendmail` does the delivery directly.

names is deferred until delivery. Forwarding is also performed as the local recipients are verified.

`sendmail` appends each name to the recipient list after parsing. When a name is aliased or forwarded, the old name is retained in the list, and a flag is set that tells the delivery phase to ignore this recipient. This list is kept free from duplicates, preventing alias loops and duplicate messages delivered to the same recipient, as might occur if a person is in two groups.

Message Collection

`sendmail` then collects the message. The message should have a header at the beginning. No formatting requirements are imposed on the message body except that they must be lines of text (in other words, binary data is not allowed). The header is stored in memory, and the body of the message is saved in a temporary file.

To simplify the program interface, the message is collected even if no names were valid. The message will be returned with an error.

Message Delivery

For each unique mailer and host in the recipient list, `sendmail` calls the appropriate mailer. Each mailer invocation sends to all users receiving the message on one host. Mailers that only accept one recipient at a time are handled properly.

The message is sent to the mailer using one of the same three interfaces used to submit a message to `sendmail`. Each copy of the message is prepended by a customized header. The mailer status code is caught and checked, and a suitable error message given as appropriate. The exit code must conform to a system standard or a generic message ("Service unavailable") is given.

Queueing for Retransmission

If the mailer returned a status that indicated that it might be able to handle the mail later, `sendmail` will queue the mail and try again later.

Return to Sender

If errors occur during processing, `sendmail` returns the message to the sender for retransmission. The letter can be mailed back or written in the `dead.letter` file in the sender's home directory⁴.

Message Header Editing

Certain editing of the message header occurs automatically. Header lines can be inserted under control of the configuration file. Some lines can be merged; for example, a "From:" line and a "Full-name:" line can be merged under certain circumstances.

⁴ Obviously, if the site giving the error is not the originating site, the only reasonable option is to mail back to the sender. Also, there are many more error disposition options, but they only effect the error message — the "return to sender" function is always handled in one of these two ways.

Configuration File

Almost all configuration information is read at runtime from a text file, encoding macro definitions (defining the value of macros used internally), header declarations (the format of header lines that it will process specially, and lines that it will add or reformat), mailer definitions (giving information such as the location and characteristics of each mailer), and name rewriting rules (a limited pattern-matching system used to rewrite names).

23.3. How to Implement and Use `sendmail`

Following flag arguments, recipient name arguments may be given, unless you run in SMTP mode. In brief, the format of recipient names is:

1. Anything in parentheses is thrown away (as a comment).
2. Anything in angle brackets (“<>”) is preferred over anything else. This rule implements the Internet standard that names of the form


```
user name <machine-name>
```

 will send to the electronic “machine-name” rather than the human “user name.”
3. Double quotes (“”) quote phrases; backslashes quote characters. Backslashes are more powerful in that they will cause otherwise equivalent phrases to compare differently — for example, `user` and `"user"` are equivalent, but `\user` is different from either of them.

Parentheses, angle brackets, and double quotes must be properly balanced and nested. The rewriting rules control the rest of processing that needs to be done.

Mail to Files and Programs

Files and programs are legitimate message recipients. Files provide archival storage of messages, useful for project administration and history. Programs are useful as recipients in a variety of situations, for example, to maintain a public repository of systems messages such as the Usenet news system.⁵

Any name passing through the initial parsing algorithm as a local name is scanned for two special cases. If prefixed by a vertical bar (“|”) the rest of the name is processed as a shell command. If the user name begins with a slash mark (“/”) the name is used as a filename, instead of a login name.

Message Collection

Once all recipient names are parsed and verified, the message is collected. The message comes in two parts: a message header and a message body, separated by a blank line.

The header is formatted as a series of lines of the form

```
field-name: field-value
```

For example, a sample header might be:

```
From: John Smith <Smith@Podunk.edu>
```

Field-value can be split across lines by starting the following lines with a space or a tab. Some header fields have special internal meaning, and have appropriate

⁵ Not provided by Sun.

special processing. Other headers are simply passed through. Some header fields may be added automatically, such as time stamps.

The body is a series of text lines. It is completely uninterpreted and untouched, except that lines beginning with a dot have the dot doubled when transmitted over an SMTP channel. This extra dot is stripped by the receiver.

Message Delivery

The send queue is grouped by receiving host before transmission to implement message batching. An argument list is built as the scan proceeds. Mail to files is detected during the scan of the send list. The interface to the mailer is performed using one of the techniques described in the section *How to Implement and Use sendmail*.

After a connection is established, `sendmail` makes the per-mailer changes to the header and sends the result to the mailer. If any mail is rejected by the mailer, a flag is set to invoke the return-to-sender function after all delivery completes.

Queued Messages

If the mailer returns a “temporary failure” exit status, the message is queued. A control file is used to describe the recipients to be sent to and various other parameters. This control file is formatted as a series of lines, each describing a sender, a recipient, the time of submission, or some other parameter of the message. The header of the message is stored in the control file so that the associated data file in the queue is just the temporary file that was originally collected.

Configuration Overview

Configuration is controlled primarily by a configuration file read at startup. Adding mailers or changing the rewriting or routing information does not require recompilation. The configuration file encodes macro definitions, header definitions, mailer definitions, rewriting rules, and options.

Macros

Macros can be used in three ways. Certain macros transmit unstructured textual information into the mail system, such as the name `sendmail` will use to identify itself in error messages. Other macros are unused internally, and can be used as shorthand in the configuration file.

Header Declarations

Header declarations inform `sendmail` of the format of known header lines. Knowledge of a few header lines is built into `sendmail`, such as the “From:” and “Date:” lines.

Most configured headers will be automatically inserted in the outgoing message if they don't exist in the incoming message. Certain headers are suppressed by some mailers.

Mailer Declarations

Mailer declarations specify the internal name of the mailer; some flags associated with the mailer, and an argument vector to be used on the call; this vector is macro-expanded before use.

Name Rewriting Rules

The heart of name parsing in `sendmail`. These are an ordered list of pattern-replacement rules, which are applied to each name. In particular, rule set zero determines which mailer to use. The name is rewritten until it is either rewritten into a special canonical form — for example, a (mailer, host, user) triple, such as {ddn, isi.edu, postel} representing the name “postel@isi.edu” — or it falls off the end. When a pattern matches, the rule is reapplied until it fails.

The configuration file also supports the editing of names into different formats. For example, a name of the form:

```
ucsfcg1!tef
```

might be mapped into:

```
tef@ucsfcg1.UUCP
```

to conform to the internal syntax. Translations can also be done in the other direction for particular mailers.

Option Setting

There are several options that can be set from the configuration file. These include the pathnames of various support files, timeouts, default modes, etc.

23.4. Basic Installation

There are two basic steps to installing `sendmail`. The first, and hardest, part is to build the configuration file. The configuration file describes the mailers it knows about, how to parse names, how to rewrite the message header, and the settings of various options. Each time `sendmail` starts up, it reads the configuration file. Although the configuration file is complex, you can usually build a configuration by adjusting an existing off-the-shelf configuration. The second step is to create and install the necessary files.

sendmail Files

You can produce your own `sendmail.cf` file from one of the generic configuration files provided with this release:
`/usr/lib/sendmail.main.cf` or
`/usr/lib/sendmail.subsidiary.cf`. If your machine is the primary mail handling machine for your site, copy `/usr/lib/sendmail.main.cf` into a file named `/etc/sendmail.cf`. For all other machines in your organization you would copy `/usr/lib/sendmail.subsidiary.cf` to `/etc/sendmail.cf`. Refer to the procedures in *Administering Electronic Mail*.

/usr/lib/sendmail

`/usr/lib/sendmail` is the actual routing program for mail.
`/usr/lib/sendmail` receives mail from user interface programs and uses the information found in the configuration files to direct the mail messages. This file contains the binary of `sendmail`.

- `/usr/lib/sendmail.mx` `/usr/lib/sendmail.mx` is the `sendmail` binary linked with the resolver; it uses `mx` records, as explained in *Administering Domain Name Service*.
- `/etc/sendmail.cf` This is the configuration file used by `/usr/lib/sendmail` to set up the appropriate environment and conditions. The configuration file can be tailored for your particular machine. Here are three subsets of the configuration file:
- `/usr/lib/sendmail.main.cf`
The default configuration file for main machines, in text form.
- `/usr/lib/sendmail.subsidiary.cf`
The default configuration file for subsidiary machines, in text form.
- `/usr/lib/sendmail.fc`
The frozen configuration file.
- `/var/spool/mqueue` The directory `/var/spool/mqueue` holds the mail queue. The default is for `/usr/lib/sendmail` to have the `setuid` bit set and `/var/spool/mqueue` to be owned by `root` with mode `755`. If this is not the case, `/var/spool/mqueue` needs to be mode `777`. Mail that has not been delivered yet is stored here. The mail messages are broken into two files: a `qf` file, which contains the control information, and the `df` file, which contains the body of the message. Delivered mail is stored in `/var/spool/mail/login_name`.
- `/var/spool/mqueue`
The directory in which the mail queue and temporary files reside.
- `/var/spool/mqueue/qf*`
Control (queue) files for messages.
- `/var/spool/mqueue/df*`
Data files.
- `/var/spool/mqueue/lf*`
Lock files
- `/var/spool/mqueue/tf*`
Temporary versions of the `qf` files, used during queue file rebuild.
- `/var/spool/mqueue/nf*`
A file used when creating a unique id.
- `/var/spool/mqueue/xf*`
A transcript of the current session.
- `/etc/aliases*` `/etc/aliases` is the optional aliases file. You can use the aliases in this file to redirect mail to another machine, send mail to a group of people, and to redirect mail from nicknames or commonly misspelled user names to the correct user. Typically, you might also use the domain-wide or "global aliases" found in the NIS database. (The global aliases are usually located in `/etc/aliases` on the master `yp` server.)

`/etc/aliases`

The textual version of the alias file. (Also referred to as “local aliases.”)

`/etc/aliases.{pag, dir}`

The alias file in dbm(3) format.

`/usr/ucb/newaliases`

The `newaliases` command updates the alias database, from the `/etc/aliases` file, which then can be found in the `/etc/aliases.dir` and `/etc/aliases.pag` files. You can adjust the configuration file to have this done automatically. This is really just a link to `/usr/lib/sendmail`. Running this program is equivalent to giving `sendmail` the `-bi` flag.

`/etc/rc.local`

The `sendmail` daemon may need to be started when your system reboots. This daemon performs two functions: it listens on the SMTP socket for connections (to receive mail from a remote system) and it processes the queue periodically. The following lines should be in `/etc/rc.local` to start up the daemons:

```
if [ -f /etc/sendmail -a -f /usr/lib/sendmail.cf ]; then
    (cd /var/spool/mqueue; rm -f nf* lf*)
    /usr/lib/sendmail -bd -qlh & echo -n ' sendmail'    >/dev/console
fi
```

The `cd` and `rm` commands ensure that all lock files have been removed; expendable lock files may be left around if the system goes down in the middle of processing a message. The line that actually invokes `sendmail` has two flags: `-bd` causes it to listen on the SMTP port for incoming mail from other machines, `-qlh` causes it to run the queue every hour. The queue on a *mailhost* is run at shorter intervals, typically every 15 minutes (the flag in this case would be `-q15m`).

`/usr/lib/sendmail.hf`

This help file is not intended to be used by the system administrator, but by the SMTP `HELP` command. The file is already installed in the distribution.

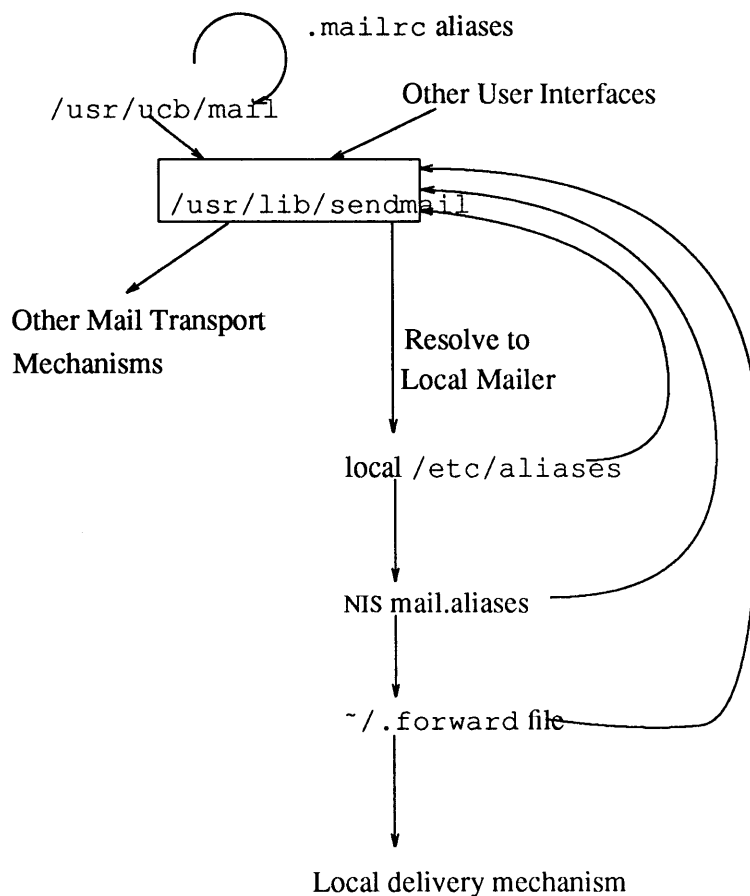
`/var/spool/mail`

Mailbox directory that can be mounted from a mailbox server using NFS.

23.5. Aliases, Forwarding, and Mailing Lists

Aliases and forwarding can be used to map one recipient name to one or more other recipients. Figure 23-2 shows how `sendmail` handles aliases. Programs that read mail, such as `/usr/ucb/mail`, can have aliases of their own, expanded before `sendmail` gets invoked.

The local alias database exists in two forms. (See the section below for information about domain-wide aliases with the NIS.) One is a text form, maintained in the file `/etc/aliases`, which is a text file.

Figure 23-2 *How sendmail Uses Aliases*

The aliases are of the form

```
name: name1, name2, ...
```

You can alias only local names. Local names are those that resolve to a particular mailer called "local," which usually implies a name with the current host name, or no host name.

For example,

```
eric@mit-xx: eric@berkeley
```

produces an error message if eric@mit-xx is not a local name. Blank lines and lines beginning with a sharp sign (" # ") are comments.

The second form is processed by the *dbm(3)* library. You can run the `/usr/ucb/newaliases` command to rebuild the binary forwarding database from the mail aliases file `/etc/aliases`, and create the dbm database files `/etc/aliases.dir` and `/etc/aliases.pag`. This is the form that sendmail actually uses to resolve aliases. This is equivalent to giving sendmail the `-bi` flag:

```
% /usr/lib/sendmail -bi
```

If the **D** option is specified in the configuration, `sendmail` rebuilds the alias database automatically if possible when it is out of date. The conditions under which it will do this are:

1. The dbm version of the database is mode 666.
2. `sendmail` is running `setuid` to root.

Auto-rebuild can be dangerous on heavily loaded machines with large alias files; if it might take more than five minutes to rebuild the database, there is a chance that several processes will start the rebuild process simultaneously.

Potential Problems

Several problems can occur with the alias database when a `sendmail` process accesses the dbm version while it is only partially built. This can happen under two circumstances. The first is when one process accesses the database while another process is rebuilding it. The second circumstance occurs when the process rebuilding the database dies (due to being killed or a system crash) before completing the rebuild.

`sendmail` has two techniques to try to relieve these problems. First, it ignores interrupts while rebuilding the database; this avoids the problem of someone aborting the process leaving a partially rebuilt database. Second, at the end of the rebuild it adds an alias of the form

```
@: @
```

which is not normally legal. Before `sendmail` accesses the database, it checks to insure that this entry exists. You need to specify the `a` option in the configuration file for this to occur. It will wait for the time specified in the `a` option for this entry to appear, at which point it will force a rebuild itself.⁶

Mailing Lists

Aliases that expand to more than one name are called mailing lists. In typical systems, most mail traffic is from lists, so take extra caution when administering them. If an error occurs when sending to a certain name, say `x`, `sendmail` will look for an alias of the form “owner-`x`” to receive the errors. This is typically useful for a mailing list where the submitter to the list has no control over the maintenance of the list itself; in this case the list maintainer would be the owner of the list. For example:

```
unix-wizards: eric@ucbarpa, wnj@monet, nosuchuser,
              sam@matisse
owner-unix-wizards: eric@ucbarpa
```

would cause “eric@ucbarpa” to get the error that will occur when someone sends to `unix-wizards` due to the inclusion of “nosuchuser” on the list.

⁶ Note: the **D** option must be specified in the configuration file for this rebuild to occur.

Inclusion

Inclusion is specified with the syntax:

```
:include: pathname
```

A name of this form reads the file specified by *pathname* and sends to all users listed in that file. The intent is *not* to directly use this feature, but rather to use this as an alias. For example, an alias of the form:

```
project: :include:/usr/project/userlist
```

is a method of letting a project maintain a mailing list without interaction with the system administrator, even if the alias file is protected.

It is not necessary to rebuild the alias database when an `:include:` list is changed.

NIS Aliases

Mail aliases can also be obtained from a domain-wide database using NIS. The NIS database is usually the `/etc/aliases` file from some central mail machine for the group or department. This database is searched after local aliases but before `forward` files. The NIS map name can be specified with the `Y` configuration option; the default name is `mail.aliases`. This feature can be disabled by turning off NIS, by not having the given map in the default domain, or by including a `Y` option with a null string as its argument.

Update Policy

As system administrator, you need to decide on a policy for updating aliases. The proper way to change forwarding might be to mail a message to “`aliases@mailhost`” that tells them the machine on which you wish to get your mail. Machines should run NIS to get automatic access to the latest alias database, instead of forcing users to manage their own forwarding, which can create problems. For example, in previous releases of `sendmail`, some users tried to forward their mail from `user@hosta` to `user@hostb` by making `hosta` an alias for `hostb` in the host file. This caused mail to go into a black hole—a problem which was fixed in Release 4.0. Another common mistake is to put a `.forward` file in the home directory of `hosta` that says “`user@hostb`.” The problem is that when the mail gets to `hostb`, it looks up “`user`” in the NIS aliases and sends it back to `user@hosta`, resulting in a loop and more bounced mail.

Naming Conventions

It is a good idea to follow a standard convention for naming users. For example, give every user an alias consisting of their first initial followed by last name. These aliases exist on the machine that relays mail to the outside world, and are used as the domain-wide `mail.aliases` map. This way, on any machine that uses domain-wide NIS aliases, you can just mail to the first initial and last name instead of having to remember the specific user and host name.

Potential Problems

Extra care must be taken to avoid loops and inconsistent databases when both local and domain-wide aliases are used. For example, consider a user named “`smith`” who moves from a machine called “`a`” to a machine named “`b`.” You might be tempted to do this by adding a local alias on machine `a` that forwards `smith@a` to `smith@b`. But if `smith` maps to `smith@a` in the domain-wide database, a loop will result.

CAUTION `:include:` files and program aliases in domain aliases may not have the expected result, unless the appropriate files are available everywhere in the domain. The domain can include aliases to a central machine which can have the files.

Forwarding

After aliasing, recipients that are local and valid are checked for the existence of a `.forward` file in their home directory. If it exists, the message is *not* sent to that user, but rather to the list of users in that file. Often this list will contain only one name, and the feature will be used for network mail forwarding.

Forwarding also permits you to specify a private program for incoming mail. For example, forwarding to:

```
"|/usr/local/newmail login_name"
```

will pipe the message into the given program instead of delivering to the default file `/var/spool/mail/login_name`. This feature is used by the vacation program.

The Mail Queue

Under conditions of high load or temporary failures, `sendmail` places a message into a job queue instead of delivering it immediately. The mail queue should be processed automatically. However, sometimes you may have to intervene manually. For example, if a major host is down for a period of time the queue may become clogged. Although `sendmail` ought to recover gracefully when the host comes up, you may find performance unacceptably bad in the meantime.

Printing the Queue

The contents of the queue can be printed by specifying the `-bp` flag to `sendmail`:

```
% /usr/lib/sendmail -bp | more
```

This produces a listing of the queue IDs, the size of the message, the date the message entered the queue, and the sender and recipients.

Format of Queue Files

All queue files located in `/var/spool/mqueue` have the form `xA999999` where `AA99999` is the *id* for this file and the *x* is a type. The types are:

- d The data file. The message body (excluding the header) is kept in this file.
- l The lock file. If this file exists, the job is currently being processed, and a queue run will not process the file. For that reason, an extraneous `lf` file can cause a job to apparently disappear.
- n This file is created when an ID is created. It is a separate file that insures that no mail can ever be destroyed due to a race condition. It should exist for no more than a few milliseconds at any given time.
- q The queue control file. This file contains the information necessary to process the job.
- t A temporary file. These are an image of the `qf` file when it is being rebuilt. It should be renamed to a `qf` file almost immediately.

- x A transcript file, existing during the life of a session showing everything that happens during that session.

The `qf` file is structured as a series of lines each beginning with a code letter. The lines are as follows:

- P The current message priority. This is used to order the queue. Higher numbers mean lower priorities. The priority increases as the message sits in the queue. The initial priority depends on the message class and the size of the message.
- T The job creation/submission time in seconds. This is used to compute when to time out the job.
- D The name of the data file. There may only be one of these lines.
- M A message. This line is printed by using `sendmail` with the `-bp` flag, and is generally used to store status information. It can contain any text.
- S The sender name. There may only be one of these lines.
- E Error recipient name. Error messages will be sent to this user instead of the sender. It is optional.
- H A header definition. There may be any number of these lines. The order is important: they represent the order in the final message. These use the same syntax as header definitions in the configuration file.
- R A recipient name. There may any number of these lines. This will normally be completely aliased, but is actually realiaed when the job is processed. There will be one line for each recipient. The recipient name must be at the end of the `qf` file.

As an example, the following is a queue file sent to “`jg@granite`”.

```
P4579
T546644595
DdfAA00393
MDeferred: Host granite.dec.com is down
S<dshr@devnull>
Hreceived: from snail.sun.com by Sun.COM (4.0/SMI-3.2)
      id AA00393; Tue, 28 Apr 87 14:43:15 PDT
Hreceived: from devnull.sun.uucp by snail.sun.com (3.2/SMI-3.2)
      id AA12023; Tue, 28 Apr 87 14:42:54 PDT
Hreceived: by devnull.sun.uucp (3.2/SMI-3.2)
      id AA01475; Tue, 28 Apr 87 14:43:37 PDT
Hmessage-id: <8704282143.AA01475@devnull.sun.uucp>
Hdate: Tue 28 Apr 1987 14:42:49 EST
HFrom: David Rosenthal <dshr@devnull>
Hsubject: X11 symlink
HTo: jg@granite.DEC.COM (Jim Gettys)
Hin-reply-to: jg's message of Tue, 28 Apr 87 10:28:17 PDT
R<jg@granite.DEC.COM>
```

This shows the name of the data file, the person who sent the message, the creation/submission time (in seconds since January 1, 1987), the message

priority, headers for the message, and the recipients.

Forcing the Queue

`sendmail` should run the queue automatically at intervals. The algorithm is to read and sort the queue, and then to attempt to process all jobs in order. When it attempts to run the job, `sendmail` first checks to see if the job is locked. If so, it ignores the job.

There is no attempt to insure that only one queue processor exists at any time, since there is no guarantee that a job cannot take forever to process. Due to the locking algorithm, it is impossible for one job to freeze the queue. However, an uncooperative recipient host or a program recipient that never returns can accumulate many processes in your system. Unfortunately, there is no way to resolve this without setting a time limit.

In some cases, you may find that a major host going down for a couple of days may create a prohibitively large queue. This will result in `sendmail` spending an inordinate amount of time sorting the queue. This situation can be fixed by moving the queue to a temporary place and creating a new queue. The old queue can be run later when the offending host returns to service.

To do this, it is acceptable to move the entire queue directory as follows:

```
# cd /var/spool
# mv mqueue omqueue; mkdir mqueue; chmod 777 mqueue
# ps ax | grep sendmail
```

Kill the existing daemon (since it will still be processing in the old queue directory) and create a new daemon. To do this, do the following:

```
# kill process_id
# /usr/lib/sendmail -bd -qlh
```

To run the old mail queue, run the following command:

```
# /usr/lib/sendmail -oQ/var/spool/omqueue -q
```

The `-oQ` flag specifies an alternate queue directory and the `-q` flag says to just run every job in the queue. Use the `-v` flag if you want to see the verbose output displayed on the screen.

When the queue is finally emptied, you can remove the directory:

```
# rmdir /var/spool/omqueue
```

You can also run a subset of the queue at any time with the `-Rstring` (run queue where any recipient name matches *string*) or with `-Mnnnnn` (run just one message, with queue id *nnnnn*). For example, you could give the command

```
% /usr/lib/sendmail -Rwnj
```

to run the subset of the queue that contains the recipient "wnj".

23.6. Arguments

The complete list of arguments to `sendmail` is described in detail in the section *Command Line Arguments*. Some important arguments are described here.

Queue Interval

The amount of time between forking a process to run through the queue is defined by the `-q` flag. If you run in mode `i` or `b` (the default) this can be relatively large, since it will only be relevant when a host that was down comes back up. If you run in `q` mode it should be relatively short, since it defines the maximum amount of time that a message may sit in the queue.

Daemon Mode

If you allow incoming mail over an TCP connection, you should have a daemon running. This should be set by your `/etc/rc.local` file using the `-bd` flag. You can combine the `-bd` flag and the `-q` flag in one call:

```
# /usr/lib/sendmail -bd -q30m
```

Debugging

There are a fairly large number of debug flags built into `sendmail`. Each debug flag has a number and a level, where higher levels indicate "print more information." The convention is that levels greater than nine are unnecessary unless you are debugging that particular piece of code. Debug flags are set using the `-d` option; the syntax is:

```
debug-flag:      -d debug-list
debug-list:      debug-option [ , debug-option ]
debug-option:    debug-range [ . debug-level ]
debug-range:     integer | integer — integer
debug-level:     integer
```

where spaces are for reading ease only. For example:

```
-d12      Set flag 12 to level 1
-d12.3    Set flag 12 to level 3
-d3-17    Set flags 3 through 17 to level 1
-d3-17.4  Set flags 3 through 17 to level 4
```

There are numerous debugging flags available for `sendmail`. If you have source code, you can refer to the list of debug flags in the code.

Trying a Different Configuration File

You can specify an alternative configuration file by using the `-C` flag; for example,

```
% /usr/lib/sendmail -Ctest.cf
```

uses the configuration file `test.cf` instead of the default `/etc/sendmail.cf`. If the `-C` flag has no value it defaults to `sendmail.cf` in the current directory.

Quick Configuration Startup

If you are using a `sendmail` command-line interface heavily for mail you can set up a fast version of the configuration file by using the `-bz` flag:

```
# /usr/lib/sendmail -bz
```

This creates the file `/etc/sendmail.fc` (“frozen configuration”). This file is an image of `sendmail`’s data space after reading in the configuration file. If this file exists, it is used instead of `/etc/sendmail.cf`. `sendmail.fc` must be rebuilt manually every time you change `sendmail.cf`.

The frozen configuration file will be ignored if a `-C` flag is specified. The frozen configuration file depends on the specific version of the `/usr/lib/sendmail` binary file. If the binary file does not match the frozen configuration file, then the frozen file will be removed. There is no checking to make sure that the frozen file has been updated after the text version of the file; it is important for you to remove the frozen file or run `/etc/sendmail -bz` after every change to the configuration file.

CAUTION

Note that once you create the frozen configuration file, the default configuration file is ignored. You need to remove `sendmail.fc` for the default configuration file to be read instead of the frozen configuration file.

23.7. Tuning

There are several configuration parameters you may want to change, depending on the requirements of your site. Most of these are set using an option in the configuration file. For example, the line `OT3d` sets option `T` to the value “3d” (three days).

Time Values

All time intervals are given using a syntax of numbers and letters. For example, “10m” represents ten minutes, whereas “2h30m” represents two and a half hours. The full set of letters is:

```
s  seconds
m  minutes
h  hours
d  days
w  weeks
```

Queue Interval

The argument to the `-q` flag specifies how often `sendmail` runs the queue. This is typically set to between fifteen minutes (`-q15m`) and one hour (`-q1h`).

Read Timeouts

It is possible to time out when reading the standard input or when reading from a remote SMTP server. Technically, this is not acceptable within the published protocols. However, it might be appropriate to set it to something large in certain environments (such as an hour). This will reduce the chance of several idle daemons piling up on your system. This timeout is set using the `r` option in the configuration file.

Message Timeouts

After sitting in the queue for a few days, a message should be returned. This is to insure that at least the sender is aware of the inability to send a message. The timeout is typically set to three days. This timeout is set using the **T** option in the configuration file.

You can flush messages that have been hanging for a short period by running the queue with a short message timeout. For example,

```
% /usr/lib/sendmail -oT1d -q
```

will run the queue and flush anything that is one day old or older.

Delivery Mode

There are several delivery modes that `sendmail` can operate in, set by the **d** configuration option. These modes specify how quickly mail will be delivered. Legal modes are:

```
i  deliver interactively (synchronously)
b  deliver in background (asynchronously)
q  queue only (don't deliver)
```

There are tradeoffs. Mode **i** passes the maximum amount of information to the sender, but is hardly ever necessary. Mode **q** puts the minimum load on your machine, but means that delivery may be delayed for up to the queue interval. Mode **b** is probably a good compromise.

Load Limiting

Often central mail machines can be overloaded. Of course the best solution is to dedicate a more powerful machine to handling mail, but load almost always expands to consume whatever resources are allocated.

The **x** and **X** options can be used to limit the load caused by `sendmail`. Both of these configuration options take an argument that is an integer load average. For example, if you specify **x4** and **X8** then the **x** load limiting will be used when the load is above four, and the **X** load limiting will be used when the load is above eight. When the load is above the value specified in the **X** option, the SMTP server will not accept connections from the network (locally originated mail and other mail such as UUCP are not affected). The **x** option has a more subtle effect, controlling whether messages are queued for later delivery or delivered immediately. The general idea is to always deliver 'small' messages immediately, and defer 'large' messages for delivery during off-peak periods.

The **q** option specifies the maximum size of message that will be delivered immediately. The "size" of the message includes not only the number of bytes in the message, but also penalties for large number of recipients, and for delivery attempts that were unsuccessful. The penalty per recipient is option value **y**, by default set to 1000. The penalty per delivery attempt is the option value **z**, by default set to 9000. The size limit is also dependent on current load, so that more and more messages are queued as load goes higher. If the load is one above the **x** threshold, then the limit is halved; if the load is two above the threshold, the limit is divided by three, etc. Note that this limit also applies to messages being delivered when running the queue, in contrast to earlier versions of `sendmail`.

The goal of load limiting is to prevent wasting time during loaded periods by attempting to deliver large messages, messages to many recipients, and messages to sites that have been down for a long time.

Log Level

The level of logging can be set for `sendmail`. The default using a standard configuration table is level 9. The levels are as follows:

- 0 No logging.
- 1 Major problems only.
- 2 Message collections and failed deliveries.
- 3 Successful deliveries.
- 4 Messages being deferred (due to a host being down, etc.).
- 5 Normal message queueups.
- 6 Unusual but benign incidents, for example, trying to process a locked queue file.
- 9 Log internal queue ID to external message ID mappings. This can be useful for tracing a message as it travels between several hosts.
- 12 Several messages that are basically only of interest when debugging.
- 16 Verbose information regarding the queue.

Refer to the `syslog` section in the chapter *Administering Workstations*, for more information.

File Modes

Certain files may have a number of modes. The modes depend on what functionality you want and the level of security you require.

To `setuid` or not to `setuid`?

By default, `sendmail` is `setuid` to root. At the point where it is about to `exec(2)` a mailer, it checks to see if the user ID is zero; if so, it resets the user ID and group ID to a default (set by the `u` and `g` options). (You can override this by setting the `S` flag to the mailer for mailers that are trusted and must be called as root.) However, this will cause mail processing to be accounted (using `sa(8)`) to root rather than to the user sending the mail.

Temporary File Modes

The mode of all temporary files that `sendmail` creates is determined by the `F` option. Reasonable values for this option are `0600` and `0644`. If the more permissive mode is selected, it will not be necessary to run `sendmail` as root at all (even when running the queue), but will allow users to read mail in the queue.

Should my Alias Database be Writable?

One approach is to provide the alias database (`/etc/aliases*`) with mode `666`. There are some dangers inherent in this approach: any user can add himself or herself to any list, or can cause any user's mail to be forwarded to another location.

23.8. The Configuration File

This section describes the configuration file in detail, including hints for writing your own, if required.

The syntax of the configuration file is designed to be reasonably easy to parse, since parsing can be done every time `sendmail` starts up. Unfortunately, this can sacrifice readability.

`sendmail` uses single letters for several different functions:

- Command-line flags
- Configuration options
- Queue file line types
- Configuration file line types
- Mailer field names
- Mailer flags
- Macro names
- Class names

This section provides an overview of the configuration file is provided, plus details of its semantics.

Building a Configuration File from Scratch

Building a configuration file from scratch is a complex task. Fortunately, it is almost never necessary to do so; nearly every situation that may come up may be resolved by changing an existing file. In any case, it is critical that you understand what it is that you are trying to do and come up with a policy statement for the delivery of mail. This section is intended to explain what the real purpose of a configuration file is and to give you some ideas for what your policy might be.

Purpose of the Configuration File

The configuration file has three major purposes. The first and simplest is to set up the environment for `sendmail`. This involves setting the options and defining a few critical macros.

The second purpose is to map names into the actual set of commands necessary to get the message delivered. Ruleset zero must resolve to the internal form, which is in turn used as a pointer to a mailer descriptor. The mailer descriptor describes the interface requirements of the mailer.

The third purpose is to rewrite names in the message. This should typically be done in two phases. The first phase maps names in any format into a canonical form. This should be done in ruleset three. The second phase maps this canonical form into the syntax appropriate for the receiving mailer. `sendmail` does this in three subphases. Rulesets one and two are applied to all sender and recipient names respectively. After this, you may specify per-mailer rulesets for both sender and recipient names; this allows mailer-specific customization. Finally, ruleset four is applied to do any conversion to external form.

RFC 822 describes the format of the mail message itself. `sendmail` follows this RFC closely, to the extent that many of the standards described in this document cannot be changed without changing the code. In particular, the following

characters have special interpretations:

```
< > ( ) " \
```

Any attempt to use these characters for other than their RFC 822 purpose in names is probably doomed to disaster.

Domains and Policies

RFC 819 describes domain-based naming. This is touched on in RFC 822 as well. Essentially each host is given a name that is a right-to-left dot qualified pseudo-path from a distinguished root. The elements of the path are organizational entities, not physical networks.

How to Proceed

Once you have decided a policy, it is worth examining the available configuration files to decide if any of them are close enough to steal major parts of. Even under the worst of conditions, there is a fair amount of boilerplate that can be collected safely.

The next step is to build ruleset three. This will be the hardest part of the job. Beware of doing too much to the name in this ruleset, since anything you do will reflect through to the message. In particular, stripping of local domains is best deferred, since this can leave you with names with no domain spec at all. Since `sendmail` likes to append the sending domain to names with no domain, this can change the semantics of names. Also try to avoid fully qualifying domains in this ruleset. Although technically legal, this can lead to unpleasantly and unnecessarily long names reflected into messages. The SunOS configuration files define ruleset nine to qualify domain names and strip local domains. This is called from ruleset zero to get all names into a cleaner form.

Once you have ruleset three finished, the other rulesets should be relatively simple. If you need hints, examine the supplied configuration files.

Testing the Rewriting Rules — the `-bt` Flag

When you build a configuration file, you can do a certain amount of testing using the “test mode” of `sendmail`. For example, you could invoke `sendmail` as:

```
% sendmail -bt -Ctest.cf
```

which would read the configuration file `test.cf` and enter test mode. For example:

```
ADDRESS TEST MODE
Enter <ruleset> <name>
>
```

In this mode, you enter lines of the form:

```
rwset name
```

where *rwset* is the rewriting set you want to use and *name* is a name to apply the set to. Test mode shows you the steps it takes as it proceeds, finally showing you the name it ends up with. You may use a comma separated list of *rwsets* for sequential application of rules to an input; ruleset three is always applied first. For example:

```
> 1,21,4 monet:bollard
```

first applies ruleset three to the input "monet:bollard." Ruleset one is then applied to the output of ruleset three, followed similarly by rulesets twenty-one and four.

If you need more detail, you can also use the `-d21` flag to turn on more debugging. For example,

```
% sendmail -bt -d21.99
```

turns on an incredible amount of information; a single word name may result in several pages worth of information.

A Sample sendmail Configuration File

```
#####
#
# Sendmail configuration file for "MAIN MACHINES"
#
# You should install this file as /etc/sendmail.cf
# if your machine is the main (or only) mail-relaying
# machine in your domain. Then edit the file to
# customize it for your network configuration.
#
# See the manual "System and Network Administration for the Sun
# Workstation". Look at "Setting Up The Mail Routing System" in
# the chapter on Communications. The Sendmail reference in the
# back of the manual is also useful.
#
# @(#)main.mc 1.16 88/09/21 SMI
#

### local info

# my official hostname
# You have two choices here. If you want the gateway machine to identify
# itself as the DOMAIN, use this line:
Dj$m
# If you want the gateway machine to appear to be INSIDE the domain, use:
#Dj$w.$m
# Unless you are using sendmail.mx (or have a fully-qualified hostname), use:
#Dj$w

# major relay mailer - typical choice is "ddn" if you are on the
# Defense Data Network (e.g. Arpanet or Milnet)
DMsmartuucp

# major relay host: use the $M mailer to send mail to other domains
DR ddn-gateway
CR ddn-gateway

# If you want to pre-load the "mailhosts" then use a line like
```

```

# Fs /usr/lib/mailhosts
# and then change all the occurrences of $%y to be $=S instead.
# Otherwise, the default is to use the hosts.byname map if NIS
# is running (or else the /etc/hosts file if no NIS).

# valid top-level domains (default passes ALL unknown domains up)
Ct arpa com edu gov mil net org
Ct us de fr jp kr nz il uk no au fi nl se ca ch my dk ar

# options that you probably want on a mailhost:

# checkpoint the queue after this many recipients
oc10

# refuse to send tiny messages to more than these recipients
Ob10

#####
#
#   General configuration information

# local domain names
#
# These can now be set from the domainname system call.
# If your NIS domain is different from the domain name you would like to have
# appear in your mail headers, edit them to be your mail domain name.
# Note that the first component of the NIS domain name is stripped off unless
# it begins with a dot or a plus sign.
# DmPodunk.EDU
# CmPodunk.EDU
#
# The Dm value is what is set in outgoing mail.  The Cm value is what is
# accepted in incoming mail.  usually these are the same, but you might
# want to have more than one Cm line to recognize more than one domain
# name during a transition.

# known hosts in this domain are obtained from gethostbyname() call

# Version number of configuration file
DVSMI-4.1

###   Standard macros

# name used for error messages
DnMailer-Daemon
# UNIX header format
DlFrom $g $d
# delimiter (operator) characters
Do.:%@!^=/[ ]
# format of a total name
Dq$g$?x ($x)$
# SMTP login message

```

```
De$j Sendmail $v/$V ready at $b

### Options

# Remote mode - send through server if mailbox directory is mounted
OR
# location of alias file
OA/etc/aliases
# default delivery mode (deliver in background)
Odbackground
# rebuild the alias file automagically
OD
# temporary file mode -- 0600 for secure mail, 0644 for permissive
OF0600
# default GID
Ogl
# location of help file
OH/usr/lib/sendmail.hf
# log level
OL9
# default messages to old style
Oo
# Cc my postmaster on error replies I generate
OPPostmaster
# queue directory
OQ/usr/spool/mqueue
# read timeout for SMTP protocols
Or15m
# status file -- none
OS/etc/sendmail.st
# queue up everything before starting transmission, for safety
Os
# return queued mail after this long
OT3d
# default UID
Oul

### Message precedences
Pfirst-class=0
Pspecial-delivery=100
Pjunk=-100

### Trusted users
T root daemon uucp

### Format of headers
H?P?Return-Path: <$g>
HReceived: $?sfrom $s $.by $j ($v/$V)
    id $i; $b
H?D?Resent-Date: $a
H?D?Date: $a
H?F?Resent-From: $q
H?F?From: $q
```



```

H?x?Full-Name: $x
HSubject:
H?M?Resent-Message-Id: <$t.$i@$j>
H?M?Message-Id: <$t.$i@$j>
HErrors-To:

#####
###   Rewriting rules   ###
#####

#   Sender Field Pre-rewriting
S1
#   None needed.

#   Recipient Field Pre-rewriting
S2
#   None needed.

#   Name Canonicalization

#   Internal format of names within the rewriting rules is:
#   anything<@host.domain.domain...>anything
#   We try to get every kind of name into this format, except for local
#   names, which have no host part.  The reason for the "<>" stuff is
#   that the relevant host name could be on the front of the name (for
#   source routing), or on the back (normal form).  We enclose the one that
#   we want to route on in the <>'s to make it easy to find.
#
S3

#   handle "from:<>" special case
R$*<>*$*      @$@          turn into magic token

#   basic textual canonicalization
R$*<$+>*$*    $2          basic RFC822 parsing

#   make sure <@a,@b,@c:user@d> syntax is easy to parse -- undone later
R@$+,$+:$+    @$1:$2:$3    change all ",", " to ":"
R@$+:$+       @$>6<@$1>:$2    src route canonical

R$+:$*;$+     @$1:$2;@$3    list syntax
R$+@$+        $:$1<@$2>     focus on domain
R$+<$+@$+>    $1$2<@$3>     move gaze right
R$+<@$+>      @$>6$1<@$2>    already canonical

#   convert old-style names to domain-based names
#   All old-style names parse from left to right, without precedence.
R$-!$+       @$>6$2<@$1.uucp>    uucphost!user
R$-.$+!$+    @$>6$3<@$1.$2>     host.domain!user
R$+%$+       @$>3$1@$2         user%host

#   Final Output Post-rewriting

```

```

S4
R$+<@$.uucp>          $2!$1                u@h.uucp => h!u
R$+                   $: $>9 $1           Clean up addr
R$*<$+>$*             $1$2$3              defocus

# Clean up an name for passing to a mailer
# (but leave it focused)
S9
R$=w!@                $@$w!$n
R@                    $@$n                handle <> error addr
R$*<$*LOCAL>$*        $1<$2$m>$3          change local info
R<@$+>$*:$+:$+       <@$1>$2,$3:$4       <route-addr> canonical

#####
# Rewriting rules

# special local conversions
S6
R$*<@$*$=m>$*        $1<@$2LOCAL>$4       convert local domain

# Local and Program Mailer specification

Mlocal, P=/bin/mail, F=rlsDFMmnP, S=10, R=20, A=mail -d $u
Mprog, P=/bin/sh, F=lsDFMeuP, S=10, R=20, A=sh -c $u

S10
# None needed.

S20
# None needed.

#####
#####
##### Ethernet Mailer specification
#####
##### Messages processed by this configuration are assumed to remain
##### in the same domain. This really has nothing particular to do
##### with Ethernet - the name is historical.

Mether, P=[TCP], F=msDFMucX, S=11, R=21, A=TCP $h
S11
R$*<@$+>$*          $@$1<@$2>$3           already ok
R$+                 $@$1<@$w>           tack on our hostname

S21
# None needed.

#####
# General code to convert back to old style UUCP names

```

```

S5
R$+<@LOCAL>      $@ $w!$1          name@LOCAL => sun!name
R$+<@$-.LOCAL>   $@ $2!$1          u@h.LOCAL => h!u
R$+<@$+.uucp>    $@ $2!$1          u@h.uucp => h!u
R$+<@$*>        $@ $2!$1          u@h => h!u
# Route-addr's do not work here.  Punt til uucp-mail comes up with something.
R<@$+>$*         $@ @$1$2          just defocus and punt
R$*<$*>$*        $@ $1$2$3         Defocus strange stuff

```

```
# UUCP Mailer specification
```

```
Muucp, P=/usr/bin/uux, F=msDFMhuU, S=13, R=23,
A=uux - -r -a$f $h!rmail ($u)
```

```
# Convert uucp sender (From) field
```

```

S13
R$+              $:>$5$1          convert to old style
R$=w!$+          $2              strip local name
R$+              $:$w!$1         stick on real host name

```

```
# Convert uucp recipient (To, Cc) fields
```

```

S23
R$+              $:>$5$1          convert to old style

```

```
#####
```

```

#
#       DDN Mailer specification
#
#       Send mail on the Defense Data Network
#       (such as Arpanet or Milnet)

```

```
Mddn, P=[TCP], F=msDFMucx, S=22, R=22, A=TCP $h, E=
```

```
# map containing the inverse of mail.aliases
```

```
DZmail.byaddr
```

```

S22
R$*<@LOCAL>$*    $:$1
R$-<@$->          $:>$3${Z$1@Z$2$}  invert aliases
R$*<@$+.$*>$*    $@$1<@$2.$3>$4    already ok
R$+<@$+>$*       $@$1<@$2.$m>$3    tack on our domain
R$+              $@$1<@$m>         tack on our domain

```

```
# "Smart" UUCP mailer: Uses UUCP transport but domain-style naming
```

```
Msmartuucp, P=/usr/bin/uux, F=CmsDFMhuU, S=22, R=22,
A=uux - -r $h!rmail ($u)
```

```
#####
```

```

#
#       RULESET ZERO

```

```

#
# This is the ruleset that determines which mailer a name goes to.

# Ruleset 30 just calls rulesets 3 then 0.
S30
R$*      $: $>3 $1          First canonicalize
R$*      @$ $>0 $1          Then rerun ruleset 0

S0
# On entry, the address has been canonicalized and focused by ruleset 3.
# Handle special cases.....
R@      $#local $:$n          handle <> form

# resolve the local hostname to "LOCAL".
R$*<*$=$w.LOCAL>$*      $1<$2LOCAL>$4          thishost.LOCAL
R$*<*$=$w.uucp>$*      $1<$2LOCAL>$4          thishost.uucp
R$*<*$=$w>$*          $1<$2LOCAL>$4          thishost

# Mail addressed explicitly to the domain gateway (us)
R$*<@LOCAL>      @$>30$1          strip our name, retry
R<@LOCAL>:$+      @$>30$1          retry after route strip

# For numeric spec, you can't pass spec on to receiver, since old rcvr's
# are not smart enough to know that [x.y.z.a] is their own name.
R<@[$+]>:$*      $:$>9 <@[$1]>:$2          Clean it up, then...
R<@[$+]>:$*      $#ether @$[$1] $:$2          numeric internet spec
R<@[$+]>,$*      $#ether @$[$1] $:$2          numeric internet spec
R$*<@[$+]>      $#ether @$[$2] $:$1          numeric internet spec

# deliver to known ethernet hosts explicitly specified in our domain
R$*<@$%y.LOCAL>$* $#ether @$2 $:$1<@$2>$3          user@host.sun.com

# etherhost.uucp is treated as etherhost.$m for now.
# This allows them to be addressed from uucp as foo!sun!etherhost!user.
R$*<@$%y.uucp>$*      $#ether @$2 $:$1<@$2>$3          user@etherhost.uucp

# Explicitly specified names in our domain -- that we've never heard of
R$*<@$*.LOCAL>$*      $#error $:Never heard of host $2 in domain $m

# Clean up addresses for external use -- kills LOCAL, route-addr ,=>:
R$*      $:$>9 $1          Then continue...

# resolve UUCP-style names
R<@[$-.uucp]>:$+      $#uucp  @$1 $:$2          @host.uucp:...
R$*<@[$-.uucp]>      $#uucp  @$2 $:$1          user@host.uucp

# Pass other valid names up the ladder to our forwarder
#R$*<@$*.$=T>$*      $#M      @$R $:$1<@$2.$3>$4 user@domain.known

# Replace following with above to only forward "known" top-level domains
R$*<@$*.$+>$*      $#M      @$R $:$1<@$2.$3>$4 user@any.domain

# if you are on the DDN, then comment-out both of the the lines above

```

```

# and use the following instead:
#R$*<@$*.$+>$*      $#ddn $@ $2.$3 $:$1<@$2.$3>$4      user@any.domain

# All addresses in the rules ABOVE are absolute (fully qualified domains).
# Addresses BELOW can be partially qualified.

# deliver to known ethernet hosts
R$*<@$%y>$*        $#ether $@ $2 $:$1<@$2>$3      user@etherhost

# other non-local names have nowhere to go; return them to sender.
R$*<@$+.$->$*      $#error $:Unknown domain $3
R$*<@$+>$*        $#error $:Never heard of $2 in domain $m
R$*@$*            $#error $:I don't understand $1@$2

# Local names with % are really not local!
R$+%$+           @$>30$1@$2          turn % => @, retry

# everything else is a local name
R$+              $#local $:$1        local names

```

Configuration File Syntax

The configuration file is organized as a series of lines, each of which begins with a single character defining the semantics for the rest of the line. Lines beginning with a space or a tab are continuation lines (although the semantics are not well defined in many places). Blank lines and lines beginning with a pound sign (#) are comments.

D — Define Macro

Macros are named with a single character. You can select these from the entire ASCII set, but you should select user-defined macros from the set of uppercase letters only. Lowercase letters and special symbols are used internally.

The syntax for macro definitions is:

Dxval

where *x* is the name of the macro and *val* is the value it should have. Macros can be inserted in most places using the escape sequence `$x`.

Following is an example of a macro definition from the configuration file:

```
DJ$w, $M
```

C and F — Define Classes

You can define classes of words to match on the left hand side of rewriting rules. For example, you might create a class of all local names for this site so that you can eliminate attempts to send to yourself.

These can either be defined directly in the configuration file or read in from another file or from another command. You can give classes names from the set of uppercase letters. Lowercase letters and special characters are reserved for system use.

The syntax is:

```
Cc word1 word2
Fc file
Fc | command
```

The first form defines the class *c* to match any of the named words. The second form reads words from the file into the class *c*, for example, `Fc /etc/hosts`. The format, if given, is used with `scanf` to read from the file; otherwise, the first word from each line is used. The third form executes the given command and reads the elements of the class from standard output of the command, for example:

```
FC | awk '{print $2}' /etc/hosts
```

You could split all these class lines among multiple lines; for example, the two forms:

```
CHmonet ucbmonet
```

and

```
CHmonet
CHucbmonet
```

are equivalent.

Macro and class names can be any letter, with lowercase reserved for internal use, uppercase for users.

O — Set Option

There are several options (not to be confused with mailer flags or command line arguments) that can be set from a configuration file. Options are also represented by single characters. The syntax of this line is:

```
Oovalue
```

This sets option *o* to *value*. Depending on the option, *value* may be a string, an integer, a boolean (with legal values “t,” “T,” “f,” or “F” — the default is TRUE), or a time interval. See the section *Configuration Options* for the list of options.

P — Precedence Definitions

You can define values for the “Precedence:” field using the **P** control line. The syntax of this field is:

```
Pname=num
```

when the *name* is found in a “Precedence:” field, the message class is set to *num*. Higher numbers mean higher precedence. Numbers less than zero have the special property that error messages will not be returned. The default precedence is zero. For example, our list of precedences is:

```
Pfirst-class=0
Pspecial-delivery=100
Pjunk=-100
```

- T — Define Trusted Users** Trusted users are those users who are permitted to override the sender name using the `-f` flag. These typically are “root,” “uucp,” and “network,” but for some users it may be convenient to extend this list to include other users, perhaps to support a separate UUCP login for each host. The syntax of this line is:
- ```
T user1 user2 . . .
```
- There may be more than one of these lines.
- H — Define Header** The format of the header lines is defined by the **H** line. The syntax of this line is:
- ```
H[?mflags?] hname:htemplate
```
- Continuation lines in this specification are inserted directly into the outgoing message. The *htemplate* is macro expanded before it is inserted into the message. If the expansion is empty, the header line is not included. If the *mflags* (surrounded by question marks) are specified, at least one of the specified flags must be stated in the mailer definition for this header to be automatically output. If one of these headers is in the input it is directed to the output regardless of these flags.
- Special Header Lines** Several header lines have special interpretations defined by the configuration file. Others have interpretations built into `sendmail` that cannot be changed without changing the code. These builtins are described here.
- Return-Receipt-To:** If this header is sent, a message will be sent to any specified names when the final delivery is complete. The mailer must have the **I** flag (local delivery) set in the mailer descriptor
- Errors-To:** If errors occur anywhere during processing, this header will cause error messages to go to the listed names rather than to the sender. This is intended for mailing lists.
- To:** If a message comes in with no recipients listed in the message (in a `To:`, `Cc:`, or `Bcc:` line) then `sendmail` will add an “Apparently To:” header line for each recipient specified on the `sendmail` command line.
- R and S — Rewriting Rules** Address parsing is done according to the rewriting rules. These are a simple pattern-matching system. `sendmail` scans through the set of rewriting rules looking for a match on the left hand side (LHS) of the rule. When a rule matches, the name is replaced by the right hand side (RHS) of the rule.
- There are several sets of rewriting rules. Some of the rewriting sets are used internally and must have specific semantics. Other rewriting sets do not have specifically assigned semantics, and may be referenced by the mailer definitions or by other rewriting sets.
- The syntax of these two lines are:

Sn

sets the current ruleset being collected to *n*. If you begin a ruleset more than once it deletes the old definition.

Rlhs *rhs* *comments*

Following is an example of how a ruleset definition might look:

```
# handle "from:<>" special case
R<>                $@@                      turn into magic token
```

The fields must be separated by at least one tab character; there may be embedded spaces in the fields. The *lhs* is a pattern that is applied to the input. If it matches, the input is rewritten to the *rhs*. The *comments* are ignored.

M — Define Mailer

Programs and interfaces to mailers are defined on this line. The format is:

Mname, {*field=value*} *

where *name* is the name of the mailer (used in error messages) and the “field=value” pairs define attributes of the mailer. Fields are:

Path	The pathname of the mailer
Flags	Special flags for this mailer
Sender	A rewriting set for sender names
Recipient	A rewriting ruleset for recipient names
Argv	An argument vector to pass to this mailer
Eol	The end-of-line string for this mailer
Maxsize	The maximum message length to this mailer
Length	The maximum length of the argv for this mailer

Only the first character of the field name is checked.

The Semantics

This section describes the details of rewriting rules and mailer descriptions.

Special Macros, Conditionals

Macros are referenced using the construct $\$x$, where *x* is the name of the macro to be matched (LHS) or inserted (RHS). Lower case letters are reserved to have special semantics, and some special characters are reserved to provide conditionals.

The following macros *must* be defined to transmit information into `sendmail`:

e	The SMTP entry message
j	The official domain name for this site
l	The format of the UNIX "From" line
n	The name of the daemon (for error messages)
o	The set of "separators" in names
q	default format of sender names

The $\$e$ macro is printed out when SMTP starts up. The first word of $\$e$ should be the $\$j$ macro. The $\$j$ macro should be in domain name format. The $\$o$ macro consists of a list of characters which will be considered tokens and which will separate tokens when scanning. For example, if “y” were in the $\$o$ macro, then the input “xyzy” would be scanned as four tokens: “x,” “y,” and “zz” and “y”. Finally, the $\$q$ macro specifies how a sender name should appear in a

message when it is created. For example, on Sun's mail routing system these definitions are:

```
De$j Sendmail $v ready at $b
DnMAILER-DAEMON
DlFrom $g $d
Do.:%@!^=/
Dq$g$?x ($x) $.
Dj$H.$D
```

All of these macros need not be changed except under unusual circumstances.

An acceptable alternative for the `$q` macro is “`$?x$x $.<$g>`”. These correspond to the following two formats:

```
nowicki@sun.com (Bill Nowicki)
Bill Nowicki <nowicki@sun.com>
```

Some macros are defined by `sendmail` for use in mailer arguments or for other contexts. These macros are:

- a** The origination date in ARPANET format
- b** The current date in ARPANET format
- c** The hop count
- d** The date in UNIX (ctime) format
- f** The sender (from) name
- g** The sender name relative to the recipient
- h** The recipient host
- i** The queue ID
- m** The domain name
- p** Sendmail's process ID
- r** Protocol used
- s** Sender's host name
- t** A numeric representation of the current time
- u** The recipient user
- v** The version number of `sendmail`
- w** The hostname of this site
- x** The full name of the sender
- z** The home directory of the recipient

There are three types of dates that can be used. The `$a` and `$b` macros are in ARPANET format; `$a` is the time as extracted from the “Date:” line of the message (if there was one), and `$b` is the current date and time (used for postmarks). If no “Date:” line is found in the incoming message, `$a` is set to the current time also. The `$d` macro is equivalent to the `$a` macro in UNIX (ctime) format.

The `$f` macro is the ID of the sender as originally determined; when mailing to a specific host the `$g` macro is set to the name of the sender *relative to the recipient*. For example, suppose the sender “eric” sends to “bollard@matisse” from the machine “ucbarpa” the `$f` macro will be “eric” and the `$g` macro will be “eric@ucbarpa.”

The `$x` macro is set to the full name of the sender. This can be determined in several ways. It can be passed as flag to `sendmail`. The second choice is the

value of the "Full-name:" line in the header if it exists, and the third choice is the comment field of a "From:" line. If all of these fail, and if the message is being originated locally, the full name is looked up in the `/etc/passwd` file.

When sending, the `$h`, `$u`, and `$z` macros get set to the host, user, and home directory (if local) of the recipient. The first two are set from the `$@` and `$:` part of the rewriting rules, respectively.

The `$p` and `$t` macros are used to create unique strings (for example, for the "Message-Id:" field). The `$i` macro is set to the queue ID on this host; if put into the timestamp line it can be useful for tracking messages. The `$v` macro is set to be the version number of `sendmail`; this is normally put in timestamps and has been proven extremely useful for debugging. The `$w` macro is set to the primary name of this host as given by `gethostname(1)` and `gethostbyname(3)`. The `$c` field is set to the "hop count," that is, the number of times this message has been processed. This can be determined by the `-h` flag on the command line or by counting the timestamps in the message.

The `$r` and `$s` fields are set to the protocol used to communicate with `sendmail` and the sending hostname;

You can specify conditionals by using the syntax:

```
$?x text1 $| text2 $.
```

This inserts `text1` if the macro `$x` is set, and `text2` otherwise. The "else" (`$|`) clause may be omitted.

Special Classes

The class `$=w` is the set of all names this host is known by. This can be used to delete local hostnames. The class `$=m` is set to the domain name.

The Left Hand Side

The left hand side of rewriting rules contains a pattern. Normal words are simply matched directly. Dollars signs introduce "metasymbols", which match things other than simple words, such as macros or classes.

The metasymbols are:

<code>\$*</code>	Match zero or more tokens
<code>\$+</code>	Match one or more tokens
<code>\$-</code>	Match exactly one token
<code>\$=x</code>	Match any string in class <code>x</code>
<code>\$~x</code>	Match any token not in class <code>x</code>
<code>\$/x</code>	Match any token in yp map <code>\$x</code>
<code>!x</code>	Match any token not in yp map <code>\$x</code>
<code>\$x</code>	Match macro <code>x</code>

If any of these match, they are assigned to the symbol `$n` for replacement on the right hand side, where `n` is the index in the LHS. For example, if the LHS

```
$- : $+
```

is applied to the input

```
UCBARPA:eric
```

the rule will match, and the values passed to the RHS will be:

```
$1 UCBARPA
$2 eric
```

The `$%x` uses the macro `x` to specify the name of an NIS map. The special form `$%y` matches any hostname in the `hosts.byname` map, or in `/etc/hosts` if not running NIS.

The Right Hand Side

When the left hand side of a rewriting rule matches, the input is replaced by the right hand side. Tokens are copied directly from the right-hand side unless they begin with a dollar sign. Metasymbols for more complicated substitutions are:

<code>\$x</code>	Expand macro <code>x</code>
<code>\$n</code>	Substitute indefinite token <code>n</code> from LHS
<code>\$>n</code>	Call ruleset <code>n</code>
<code>##mailer</code>	Resolve to <code>mailer</code>
<code>\$@host</code>	Specify <code>host</code> (+ prefix? ruleset return)
<code>\$:user</code>	Specify <code>user</code> (+ prefix rule limit)
<code>\$(host\$)</code>	Map to primary hostname
<code>\$(x name\$)</code>	Map name through <code>yp</code> map <code>\$x</code> .

The `$n` (`n` being a digit) syntax substitutes the corresponding value from a `$+`, `$-`, `$*`, `$=`, or `$~` match on the LHS. It may be used anywhere.

The `$>n` syntax causes the remainder of the line to be substituted as usual and then passed to ruleset `n`. The final value of ruleset `n` then becomes the substitution for this rule. (This can be compared to a procedure or function call.)

The `##` syntax should *only* be used in ruleset zero. It causes evaluation of the ruleset to terminate immediately, and signals to `sendmail` that the name has completely resolved. The complete syntax is:

```
##mailer$@host$:user
```

This specifies the {`mailer`, `host`, `user`} triple necessary to direct the mailer. More processing may then take place depending on the mailer. For example, local names are aliased.

A right-hand side may also be preceded by a `$@` or a `$:` to control evaluation. A `$@` prefix causes the ruleset to return with the remainder of the right-hand side as the value. A `$:` prefix causes the rule to terminate immediately, but the ruleset to continue; thus this can be used to limit a rule to one application. Neither prefix affects the result of the right-hand side expansion.

The `$@` and `$:` prefixes can precede a `$>` spec; for example:

```
R$+    $:$>7$1
```

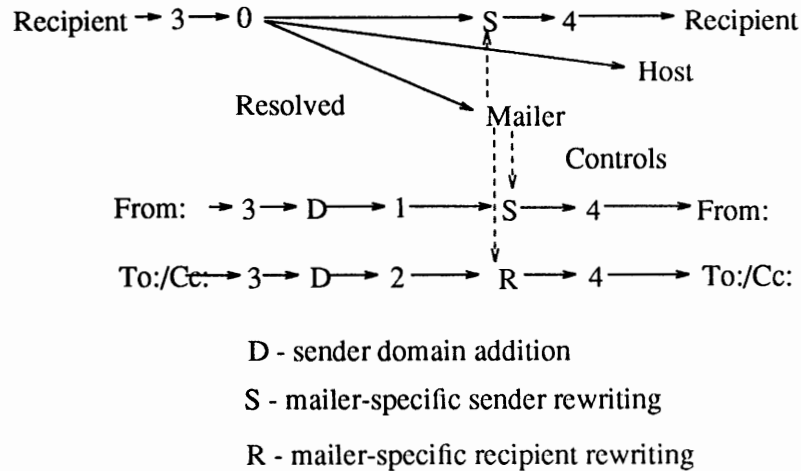
matches anything, passes that to ruleset seven, and continues; the `$:` is necessary to avoid an infinite loop. The `$(host)$` syntax replaces the host name with the "official" or primary host name, the one listed first in the `hosts.byname` `yp` map, or `/etc/hosts` if not running `yp`. This is used to eliminate "nicknames" for

hosts. The $\$(x \text{ name } \$)$ syntax replaces the string by the result of the `yp` map indicated in macro $\$x$.

Semantics of Rewriting Rule Sets

There are five rewriting sets that have specific semantics. These are related as depicted by Figure 23-3.

Figure 23-3 *Rewriting Set Semantics*



Ruleset three should turn the name into “canonical form.” This form should have the basic syntax:

```
local-part@host-domain-spec
```

If no `@` sign is specified, then the `host-domain-spec` *may* be appended from the sender name (if the `C` flag is set in the mailer definition corresponding to the *sending* mailer). Ruleset three is applied by `sendmail` before doing anything with any name.

Ruleset zero is applied after ruleset three to names that are going to actually specify recipients. It must resolve to a $\{mailer, host, user\}$ triple. The *mailer* must be defined in the mailer definitions from the configuration file. The *host* is defined into the $\$h$ macro for use in the argument expansion of the specified mailer; the *user* is defined into $\$u$.

Rulesets one and two are applied to all **from:**, **to:**, and **cc:** recipient names respectively. Then the rulesets specified in the mailer definition line (`S=` and `R=`) are applied. Note that this will be done many times for one message, depending on how many mailers the message is routed to by ruleset zero.

Ruleset four is applied last to all names in the message. It is typically used to translate internal to external form.

The “error” Mailer

The mailer with the special name “error” can be used to generate a user error. The user field is a message to be printed. For example, the entry:

```
$#error$:Host unknown in this domain
```

on the RHS of a rule will cause the specified error to be generated if the LHS matches. This mailer is only functional in ruleset zero.

Semantics of Mailer Descriptions

Each mailer has an internal name. This can be arbitrary, except that the names “local” and “prog” must be defined first and second, respectively. Ruleset zero will resolve names to this mailer name (and a host and user name).

The pathname of the mailer must be given in the P field. If this mailer should be accessed via a TCP connection, use the string “[TCP]” instead.

The F field defines the mailer flags. You should specify an “f” or “r” flag to pass the name of the sender as a `-f` or `-r` flag respectively. These flags are only passed if they were passed to `sendmail`, so that mailers that give errors under some circumstances can be placated. If the mailer is not picky you can just specify `-f$g` in the `argv` template. If the mailer must be called as `root` the “S” flag should be given; this will not reset the `userid` before calling the mailer.⁷ If this mailer is local (that is, will perform final delivery rather than another network hop) the “l” flag should be given. Quote characters (backslashes and “ marks) can be stripped from names if the “s” flag is specified; if this is not given they are passed through. If the mailer is capable of sending to more than one user on the same host in a single transaction the “m” flag should be stated. If this flag is on, then the `argv` template containing `$u` will be repeated for each unique user on a given host. The “e” flag will mark the mailer as being ‘expensive,’ which will cause `sendmail` to defer connection until a queue run.⁸

An unusual case is the “C” flag. This flag applies to the mailer that the message is received from, rather than the mailer being sent to; if set, the domain specification of the sender (that is, the “@host.domain” part) is saved and is appended to any names in the message that do not already contain a domain specification. For example, a message of the form:

```
From: eric@ucbarpa
To: wnj@monet, mckusick
```

will be modified to:

```
From: eric@ucbarpa
To: wnj@monet, mckusick@ucbarpa
```

if and only if the “C” flag is defined in the mailer corresponding to “eric@ucbarpa.”

The S and R fields in the mailer description are per-mailer rewriting sets to be applied to sender and recipient names respectively. These are applied after the sending domain is appended and the general rewriting sets (number one or two)

⁷ `sendmail` must be running `setuid` to `root` for this to work.

⁸ The `c` configuration option must be given for this to be effective.

are applied, but before the output rewrite (ruleset four) is applied. A typical use is to append the current domain to names that do not already have a domain. For example, a header of the form:

```
From: eric@host
```

might be changed to be:

```
From: eric@host.Podunk.EDU
```

or

```
From: ucbvax!eric
```

depending on the domain it is being shipped into. These sets can also be used to do special purpose output rewriting in cooperation with ruleset four.

The E field defines the string to use as an end-of-line indication. A string containing return and newline is the default, if using TCP, otherwise just a newline indicates end-of-line. You can use the `printf` backslash escapes (`\r`, `\n`, `\f`, `\b`).

An argument vector template is given as the A field. It may have embedded spaces. The template is macro-expanded before being passed to the mailer. Useful macros include `$h`, the host name resolved by ruleset zero, and `$u`, the user name (or names) resolved. If there is no argument with a `$u` macro in it, `sendmail` will use SMTP to communicate with the mailer. If the pathname for this mailer is "[TCP]," the argument vector should be

```
TCP $h [ port ]
```

where *port* is the optional port number to connect to.

If an L field exists, it specifies the maximum length of the `$u` macro passed to the mailer. This can be used with the `m` flag to send multiple recipients with one call to the mailer, while avoiding mailer limitations on argument length. This makes UUCP mail more efficient. `$u` will always expand to at least one recipient even if that recipient exceeds the L= limit.

For example, the specification:

```
Mlocal, P=/bin/mail, F=rlsmi, S=10, R=20, A=mail -d $u
Mether, P=[TCP], F=meC, S=11, R=21, A=TCP $h, M=100000
```

specifies a mailer to do local delivery and a mailer for Ethernet delivery. The first is called "local," is located in the file `/bin/mail`, takes a `-r` flag, does local delivery, quotes should be stripped from names, and multiple users can be delivered at once; ruleset ten should be applied to sender names in the message and ruleset twenty should be applied to recipient names; the argument vector to send to a message will be the word "mail," the word "-d," and words containing the name of the receiving user. If a `-r` flag is inserted it will be between the words "mail" and "-d." The second mailer is called "ether," it should be connected to via TCP, it can handle multiple users at once, connections should be deferred, and any domain from the sender name should be appended to any receiver name without a domain; sender names should be processed by ruleset eleven and recipient names by ruleset twenty-one. There is a 100,000 byte limit on messages passed through this mailer.

23.9. Command Line Arguments

Command-line arguments must appear on the `/usr/lib/sendmail` command line. These arguments are:

- `-f name` The sender's name is *name*. This flag is ignored unless the real user is listed as a "trusted user" or if *name* contains an exclamation point (because of certain restrictions in UUCP).
- `-r name` An obsolete form of `-f`.
- `-h cnt` Sets the "hop count" to *cnt*. This represents the number of times this message has been processed by `sendmail` (to the extent that it is supported by the underlying networks). *cnt* is incremented during processing, and if it reaches the value of configuration option "h," `sendmail` returns the message with an error.
- `-Fname` Sets the full name of this user to *name*.
- `-n` Do not do aliasing or forwarding.
- `-t` Read the header for "To:," "Cc:," and "Bcc:" lines, and send to everyone listed in those lists. The "Bcc:" line will be deleted before sending. Any names in the argument vector will be deleted from the send list.
- `-bx` Set operation mode to *x*. Operation modes are:
 - `m` Deliver mail (default)
 - `a` Run in arpanet mode
 - `s` Speak SMTP on input side
 - `d` Run as a daemon
 - `t` Run in test mode
 - `v` Just verify recipients
 - `i` Initialize the alias database
 - `p` Print the mail queue
 - `z` Freeze the configuration file
- `-qtime` Try to process the queued up mail. If the time is given, `sendmail` will repeatedly run through the queue at the specified interval to deliver queued mail; otherwise, it only runs once.
- `-Cfile` Use a different configuration file.
- `-dlevel` Set debugging level.
- `-oxvalue` Set configuration option *x* to the specified *value*.
- `-M` *Run given message ID from the queue.*
- `-R` *Run messages for given recipient only from the queue.*

These options are described in the section *Configuration Options*.

Several configuration options may be specified as primitive flags. These are the `c`, `e`, `i`, `m`, `T`, and `v` arguments. Also, you can specify the `f` configuration option as the `-s` argument.

23.10. Configuration Options

You can set the following options using the `-o` flag on the command line or the `O` line in the configuration file:

- Afile* Use the named *file* as the alias file instead of `/etc/aliases`. If no file is specified, use `aliases` in the current directory.
- atime* If set, time to wait for an "@:@" entry to exist in the alias database before starting up. If it does not appear after that time, rebuild the database.
- Bvalue* Blank substitute. Default is "."
- bn* Disallow empty messages to more than *n* recipients.
- Cn* Checkpoint after *n* recipients.
- c* If an outgoing mailer is marked as being expensive, do not connect immediately. This requires that a queue run processes to actually send the mail.
- D* If set, rebuild the alias database if necessary and possible. If this option is not set, `sendmail` will never rebuild the alias database unless explicitly requested using `-bi`.
- dx* Deliver in mode *x*. Legal modes are:
- i* Deliver interactively (synchronously)
 - b* Deliver in background (asynchronously)
 - q* Just queue the message (deliver during queue run)
- ex* Dispose of errors using mode *x*. The values for *x* are:
- p* Print error messages (default)
 - q* No messages, just give exit status
 - m* Mail back errors to sender
 - w* Write back errors (mail if user not logged in)
 - e* Mail back errors and give zero exit stat always
- Fn* The temporary queue file mode, in Octal. 644 and 600 are good choices.
- f* Save UNIX-style "From" lines at the front of headers. Normally they are assumed redundant and discarded.
- gn* Set the default group ID for mailers to run in to *n*.
- Hfile* Specify the help file for SMTP [Postel82].
- hn* Set maximum hop count to *n*.
- i* Ignore dots in incoming messages.
- Ln* Set the default log level to *n*.
- Mxvalue* Set the macro *x* to *value*. This is intended only for use from the command line.
- m* Send to me too, even if I am in an alias expansion.

- o Assume that the headers may be in old format, that is, spaces delimit names. This actually turns on an adaptive algorithm: if any recipient name contains a comma, parenthesis, or angle bracket, it will be assumed that commas already exist. If this flag is not on, only commas delimit names. Headers are always output with commas between the names.
- P The name of the local Postmaster. If defined, error messages from the MAILER-DAEMON cause the header to be sent to this name.
- Qdir* Use the named *dir* as the queue directory.
- qlimit* Size limit of messages to be queued under heavy load. Default is 10,000 bytes.
- Rserver* Remote mode. Deliver through remote SMTP server. Default is location of `/var/spool/mail`.
- rtime* Timeout reads after *time* interval.
- Sfile* Save statistics in the named *file*.
- s Be super-safe when running things, that is, always instantiate the queue file, even if you are going to attempt immediate delivery. `sendmail` always instantiates the queue file before returning control to the client under any circumstances.
- Ttime* Set the queue timeout to *time*. After this interval, messages that have not been successfully sent will be returned to the sender.
- un* Set the default userid for mailers to *n*. Mailers without the *S* flag in the mailer definition will run as this user.
- v Run in verbose mode.
- Xn* Set the load average value, which causes the `sendmail` daemon to refuse incoming SMTP connections to reduce system load. Default is zero, which disables this feature.
- xn* Set the load average value which causes `sendmail` to simply queue mail (regardless of the *dx* option) to reduce system load. Default is zero, which disables this feature.
- Yname* NIS map name to be used for aliases. Default is `mail.aliases`.
- yn* Recipient factor. Penalize messages with this many bytes-per-recipient.
- Zn* Time factor. Penalize messages with this many bytes-per-delivery attempts.
- zn* Message class factor. Penalize messages with this many bytes-per-class.

23.11. Mailer Flags

You can set the following flags in the mailer description.

- C If mail is *received* from a mailer with this flag set, any names in the header that do not have an at sign (“@”) after being rewritten by ruleset three will have the “@domain” clause from the sender tacked on. This allows mail with headers of the form:

```
From: usera@hosta
To: userb@hostb, userc
```

to be rewritten as:

```
From: usera@hosta
To: userb@hostb, userc@hosta
```

automatically.

- D This mailer wants a “Date:” header line.
- E Escape “From” lines to be “>From” (usually specified with U).
- e This mailer is expensive to connect to, so try to avoid connecting normally; any necessary connection will occur during a queue run.
- F This mailer wants a “From:” header line.
- f The mailer wants a `-f from` flag, but only if this is a network forward operation (that is, the mailer will give an error if the executing user does not have special permissions).
- h Uppercase should be preserved in host names for this mailer.
- L Limit the line lengths as specified in RFC 821.
- l This mailer is local (that is, final delivery will be performed).
- M This mailer wants a “Message-Id:” header line.
- m This mailer can send to multiple users on the same host in one transaction. When a `$u` macro occurs in the `argv` part of the mailer definition, that field will be repeated as necessary for all qualifying users. The `L=` field of the mailer description can be used to limit the total length of the `$u` expansion.
- n Do not insert a UNIX-style “From” line on the front of the message.
- P This mailer wants a “Return-Path:” line.
- p Always add local host name to the “MAIL From:” line of *SMTP*, even if there already is one.
- r Same as `f`, but sends a `-r` flag.
- S Do not reset the `userid` before calling the mailer. This would be used in a secure environment where `sendmail` ran as root. This could be used to avoid forged names.
- s Strip quote characters off of the name before calling the mailer.

- U This mailer wants UNIX-style “From” lines with the ugly UUCP-style “remote from <host>” on the end.
- u Uppercase should be preserved in user names for this mailer.
- X Uses the hidden dot algorithm as specified in RFC 821; basically, any line beginning with a dot will have an extra dot prepended (to be stripped at the other end). This insures that lines in the message containing a dot will not terminate the message prematurely.
- x This mailer wants a “Full-Name:” header line.

File System Check Program

When the SunOS operating system is brought up, a consistency check of the file systems should always be performed. This precautionary measure helps to insure a reliable environment for file storage on disk. If an inconsistency is discovered, corrective action must be taken. `fsck` runs in two modes. Normally it is run non-interactively by the system after a normal boot. When running in this mode, it will only make changes to the file system that are known to always be correct. If an unexpected inconsistency is found, `fsck` will exit with a non-zero exit status, leaving the system running single user. Typically you then run `fsck` interactively. When running in this mode, each problem is listed followed by a suggested corrective action. You must decide whether or not the suggested correction should be made.

The purpose of this appendix is to dispel the mystique surrounding file system inconsistencies. It first describes the updating of the file system (the calm before the storm) and then describes file system corruption (the storm). Finally, the set of deterministic corrective actions used by `fsck` is presented.

24.1. Overview of the File System

The file system is discussed in detail in [McKusick84]; this section gives a brief overview.

Superblock

A file system is described by its *superblock*. The superblock is built when the file system is created (see `newfs` (8)) and never changes. The superblock contains the basic parameters of the file system, such as the number of data blocks it contains and a count of the maximum number of files. Because the superblock contains critical data, `newfs` replicates it to protect against catastrophic loss. The *default superblock* always resides at a fixed offset from the beginning of the file system's disk partition. The *redundant superblocks* are not referenced unless a head crash or other hard disk error causes the default superblock to be unusable. The redundant blocks are sprinkled throughout the disk partition.

Within the file system are files. Certain files are distinguished as directories and contain collections of pointers to files that may themselves be directories. Every file has a descriptor associated with it called an inode. The inode contains information describing ownership of the file, time stamps indicating modification and access

⁸ This document reflects the use of `fsck` with the file system organization implemented in Release 4.1 of the SunOS operating system. This is a revision of the original paper written by T. J. Kowalski and modified by Kirk McKusick.

times for the file, and an array of indices pointing to the data blocks for the file. This section assumes that the first 12 blocks of the file are directly referenced by values stored in the inode structure itself[†]. The inode structure may also contain references to indirect blocks containing further data block indices. In a file system with a 8192 byte block size, a singly indirect block contains 1024 further block addresses, a doubly indirect block contains 1024 addresses of further single indirect blocks, and a triply indirect block contains 1024 addresses of further doubly indirect blocks (the triply indirect block is never needed in practice).

The standard SunOS block size is 8K; fragment size is 1K.

In order that files with up to 2^{32} bytes require only two levels of indirection, the minimum size of a file system block is 4096 bytes. The size of file system blocks can be any power of two greater than or equal to 4096. The block size of the file system is maintained in the superblock, so it is possible for file systems of different block sizes to be accessible simultaneously on the same system. The block size must be decided when `newfs` creates the file system; the block size cannot be subsequently changed without rebuilding the file system.

Summary Information

Associated with the superblock is non-replicated *summary information*. The summary information changes as the file system is modified. The summary information contains the number of blocks, fragments, inodes, and directories in the file system.

Cylinder Groups

The file system partitions the disk into one or more areas called *cylinder groups*. A cylinder group is comprised of one or more consecutive cylinders on a disk. Each cylinder group includes inode slots for files, a *block map* describing available blocks in the cylinder group, and summary information describing the usage of data blocks within the cylinder group. A fixed number of inodes is allocated for each cylinder group when the file system is created. The current policy is to allocate one inode for each 2048 bytes of disk space; this is expected to be far more inodes than will ever be needed.

All the cylinder group bookkeeping information could be placed at the beginning of each cylinder group. However if this approach were used, all the redundant information would be on the top platter. A single hardware failure that destroyed the top platter could cause the loss of all copies of the redundant superblocks. Thus the cylinder group bookkeeping information begins at a floating offset from the beginning of the cylinder group. The offset for the $i+1$ st cylinder group is about one track further from the beginning of the cylinder group than it was for the i th cylinder group. In this way, the redundant information spirals down into the pack; any single track, cylinder, or platter can be lost without losing all copies of the superblocks. Except for the first cylinder group, the space between the beginning of the cylinder group and the beginning of the cylinder group information stores data.

[†]The actual number may vary from system to system, but is usually in the range 5-13.

Fragments

To avoid waste in storing small files, the file system space allocator divides a single file system block into one or more *fragments*. The fragmentation of the file system is specified when the file system is created; each file system block can be optionally broken into 2, 4, or 8 addressable fragments. The lower bound on the size of these fragments is constrained by the disk sector size; typically 512 bytes is the lower bound on fragment size. The block map associated with each cylinder group records the space availability at the fragment level. Aligned fragments are examined to determine block availability.

On a file system with a block size of 8192 bytes and a fragment size of 1024 bytes, a file is represented by zero or more 8192 byte blocks of data, and possibly a single fragmented block. If a file system block must be fragmented to obtain space for a small amount of data, the remainder of the block is made available for allocation to other files. For example, consider an 11000 byte file stored on a 8192/1024 byte file system. This file uses one full size block and a 3072 byte fragment. If no fragments with at least 3072 bytes are available when the file is created, a full size block is split yielding the necessary 3072 byte fragment and an unused 1024 byte fragment. This remaining fragment can be allocated to another file, as needed.

Updates to the File System

Every working day hundreds of files are created, modified, and removed. Every time a file is modified, the operating system performs a series of file system updates. These updates, when written on disk, yield a consistent file system. The file system stages all modifications of critical information; modification can either be completed or cleanly backed out after a crash. Knowing the information that is first written to the file system, deterministic procedures can be developed to repair a corrupted file system. To understand this process, you must first know the order in which the update requests were being honored.

When a user program does an operation to change the file system, such as a *write*, the data to be written is copied into an internal *in-core* buffer in the kernel. Normally, the disk update is handled asynchronously; the user process is allowed to proceed even though the data has not yet been written to the disk. The data, along with the inode information reflecting the change, is eventually written out to disk. The real disk write may not happen until long after the *write* system call has returned. Thus at any given time, the file system, as it resides on the disk, lags behind the state of the file system represented by the in-core information.

The disk information is updated to reflect the in-core information when the buffer is required for another use, when a `sync (2)` is done (at 30 second intervals) by `/usr/etc/update (8)`, or by manual operator intervention with the `sync (8)` command. If the system is halted without writing out the in-core information, the file system on the disk will be in an inconsistent state.

If all updates are done asynchronously, several serious inconsistencies can arise. One inconsistency is that a block can be claimed by two inodes. Such an inconsistency can occur when the system is halted before the pointer to the block in the old inode has been cleared in the copy of the old inode on the disk, and after the pointer to the block in the new inode has been written out to the copy of the

new inode on the disk. Here, there is no deterministic method for deciding which inode should really claim the block. A similar problem can arise with a multiply claimed inode.

The problem with asynchronous inode updates can be avoided by doing all inode deallocations synchronously. Consequently, inodes and indirect blocks are written to the disk synchronously (that is, the process blocks until the information is really written to disk) when they are being deallocated. Similarly, inodes are kept consistent by synchronously deleting, adding, or changing directory entries.

24.2. Fixing Corrupted File Systems

A file system can become corrupted in several ways. The most common of these ways are improper shutdown procedures and hardware failures.

File systems may become corrupted during an *unclean halt*. This happens when proper shutdown procedures are not observed, physically write-protecting a mounted file system, or a mounted file system is taken off-line. The most common operator procedural failure is forgetting to `sync` the system before halting the CPU.

File systems can become further corrupted if proper startup procedures are not observed, for example, not checking a file system for inconsistencies, and not repairing those inconsistencies. Allowing a corrupted file system to be used (and to be modified further) can be disastrous.

Any piece of hardware can fail at any time. Failures can be as subtle as a bad block on a disk pack, or as blatant as a non-functional disk-controller.

Detecting and Correcting Corruption

Normally you run `fsck` non-interactively. In this mode it only fixes corruptions that are expected to occur from an unclean halt. These actions are a proper subset of the actions that `fsck` takes when it is running interactively. This appendix assumes that you are running `fsck` interactively, and all possible errors can be encountered. When an inconsistency is discovered in this mode, `fsck` reports the inconsistency for you to choose a corrective action.

You can check a quiescent file system (that is, one that is unmounted and is not being written on) for structural integrity by performing consistency checks on the redundant data present on a file system. The redundant data is either read from the file system, or computed from other known values. The file system **must** be in a quiescent state when you run `fsck` because `fsck` is a multi-pass program.

The following sections discuss methods to discover inconsistencies and possible corrective actions for the cylinder group blocks, the inodes, the indirect blocks, and the data blocks containing directory entries.

Superblock Checking

The most commonly corrupted item in a file system is the summary information associated with the superblock. The summary information is prone to corruption because it is modified with every change to the file system's blocks or inodes, and is usually corrupted after an unclean halt.

The superblock is checked for inconsistencies involving file system size, number of inodes, free block count, and the free inode count. The file system size must be larger than the number of blocks used by the superblock and the number of

blocks used by the list of inodes. The file system size and layout information are the most critical pieces of information for `fsck`. While there is no way to actually check these sizes, since they are statically determined by `newfs`, `fsck` can check that these sizes are within reasonable bounds. All other file system checks require that these sizes be correct. If `fsck` detects corruption in the static parameters of the default superblock, `fsck` requests the operator to specify the location of an alternate superblock.

Free Block Checking

`fsck` checks that all the blocks marked as free in the cylinder group block maps are not claimed by any files. When all the blocks have been initially accounted for, `fsck` checks that the number of free blocks plus the number of blocks claimed by the inodes equals the total number of blocks in the file system.

If anything is wrong with the block allocation maps, `fsck` will rebuild them, based on the list it has computed of allocated blocks.

The summary information associated with the superblock counts the total number of free blocks within the file system. `fsck` compares this count to the number of free blocks it found within the file system. If the two counts do not agree, then `fsck` replaces the incorrect count in the summary information by the actual free block count.

The summary information counts the total number of free inodes within the file system. `fsck` compares this count to the number of free inodes it found within the file system. If the two counts do not agree, then `fsck` replaces the incorrect count in the summary information by the actual free inode count.

Checking the Inode State

An individual inode is not as likely to be corrupted as the allocation information. However, because of the great number of active inodes, a few of the inodes are usually corrupted.

The list of inodes in the file system is checked sequentially starting with inode 2 (inode 0 marks unused inodes; inode 1 is saved for future generations) and progressing through the last inode in the file system. The state of each inode is checked for inconsistencies involving format and type, link count, duplicate blocks, bad blocks, and inode size.

Each inode contains a mode word. This mode word describes the type and state of the inode. Inodes must be one of six types: regular inode, directory inode, symbolic link inode, special block inode, special character inode, or socket inode. Inodes may be found in one of three allocation states: unallocated, allocated, and neither unallocated nor allocated. This last state suggests an incorrectly formatted inode. An inode can get in this state if bad data is written into the inode list. The only possible corrective action is for `fsck` is to clear the inode.

Inode Links

Each inode counts the total number of directory entries linked to the inode. `fsck` verifies the link count of each inode by starting at the root of the file system, and descending through the directory structure. The actual link count for each inode is calculated during the descent.

If the stored link count is non-zero and the actual link count is zero, then no directory entry appears for the inode. If this happens, `fsck` will place the disconnected file in the `lost+found` directory. If the stored and actual link counts are non-zero and unequal, a directory entry may have been added or removed without the inode being updated. If this happens, `fsck` replaces the incorrect stored link count by the actual link count.

Each inode contains a list, or pointers to lists (indirect blocks), of all the blocks claimed by the inode. Since indirect blocks are owned by an inode, inconsistencies in indirect blocks directly affect the inode that owns it.

`fsck` compares each block number claimed by an inode against a list of already allocated blocks. If another inode already claims a block number, then the block number is added to a list of *duplicate blocks*. Otherwise, the list of allocated blocks is updated to include the block number.

If there are any duplicate blocks, `fsck` performs a partial second pass over the inode list to find the inode of the duplicated block. The second pass is needed, since without examining the files associated with these inodes for correct content, not enough information is available to determine which inode is corrupted and should be cleared. If this condition does arise (only hardware failure will cause it), then the inode with the earliest modify time is usually incorrect, and should be cleared. If this happens, `fsck` prompts the operator to clear both inodes. The operator must decide which one should be kept and which one should be cleared.

`fsck` checks the range of each block number claimed by an inode. If the block number is lower than the first data block in the file system, or greater than the last data block, then the block number is a *bad block number*. Many bad blocks in an inode are usually caused by an indirect block that was not written to the file system, a condition which can only occur if there has been a hardware failure. If an inode contains bad block numbers, `fsck` prompts the operator to clear it.

Inode Data Size

Each inode contains a count of the number of data blocks that it contains. The number of actual data blocks is the sum of the allocated data blocks and the indirect blocks. `fsck` computes the actual number of data blocks and compares that block count against the actual number of blocks the inode claims. If an inode contains an incorrect count `fsck` prompts the operator to fix it.

Each inode contains a thirty-two bit size field. The size is the number of data bytes in the file associated with the inode. The consistency of the byte size field is roughly checked by computing from the size field the maximum number of blocks that should be associated with the inode, and comparing that expected block count against the actual number of blocks the inode claims.

Checking the Data Associated with an Inode

An inode can directly or indirectly reference three kinds of data blocks. All referenced blocks must be the same kind. The three types of data blocks are: plain data blocks, symbolic link data blocks, and directory data blocks. Plain data blocks contain the information stored in a file; symbolic link data blocks contain the path name stored in a link. Directory data blocks contain directory entries. `fsck` can only check the validity of directory data blocks.

Each directory data block is checked for several types of inconsistencies. These inconsistencies include directory inode numbers pointing to unallocated inodes, directory inode numbers that are greater than the number of inodes in the file system, incorrect directory inode numbers for “.” and “..”, and directories that are not attached to the file system. If the inode number in a directory data block references an unallocated inode, then `fsck` will remove that directory entry. Again, this condition can only arise when there has been a hardware failure.

If a directory entry inode number references outside the inode list, then `fsck` will remove that directory entry. This condition occurs if bad data is written into a directory data block.

The directory inode number entry for “.” must be the first entry in the directory data block. The inode number for “.” must reference itself; e.g., it must equal the inode number for the directory data block. The directory inode number entry for “..” must be the second entry in the directory data block. Its value must equal the inode number for the parent of the directory entry (or the inode number of the directory data block if the directory is the root directory). If the directory inode numbers are incorrect, `fsck` will replace them with the correct values. If there are multiple hard links to a directory, the first one encountered is considered the real parent to which “..” should point; `fsck` recommends deletion for the subsequently discovered names.

File System Connectivity

`fsck` checks the general connectivity of the file system. If directories are not linked into the file system, then `fsck` links the directory back into the file system in the `lost+found` directory. This condition only occurs when there has been a hardware failure.

24.3. `fsck` Error Conditions Conventions

`fsck` is a multi-pass file system check program. Each file system pass invokes a different phase of the `fsck` program. After the initial setup, `fsck` performs successive phases over each file system, checking blocks and sizes, path names, connectivity, reference counts, and the map of free blocks, (possibly rebuilding it), and performs some cleanup.

Normally `fsck` is run non-interactively to *preen* the file systems after an unclean halt. While preening a file system, it will only fix corruptions that are expected to occur from an unclean halt. These actions are a proper subset of the actions that `fsck` will take when it is running interactively. Throughout this appendix many errors have several options that the operator can take. When an inconsistency is detected, `fsck` reports the error condition. If a response is required, `fsck` prints a prompt message and waits for a response. When preening most errors are fatal. For those that are expected, the response taken is noted. This appendix explains the meaning of each error condition, the possible responses, and the related error conditions.

The error conditions are organized by the *phase* of the `fsck` program in which they can occur. The error conditions that may occur in more than one phase will be discussed in initialization.

Initialization

Before a file system check can be performed, certain tables have to be set up and certain files opened. This section concerns itself with the opening of files and the initialization of tables. This section lists error conditions resulting from command line options, memory requests, opening of files, status of files, file system size checks, and creation of the scratch file. All the initialization errors are fatal when the file system is being preened.

Message

```
C option?
```

C is not a legal option to `fsck`; legal options are `-b`, `-y`, `-n`, and `-p`. `fsck` terminates on this error condition. See the `fsck(8)` manual entry for further detail.

Message

```
cannot alloc NNN bytes for blockmap
cannot alloc NNN bytes for freemap
cannot alloc NNN bytes for statemap
cannot alloc NNN bytes for lncntp
```

`fsck`'s request for memory for its virtual memory tables failed. This should never happen. `fsck` terminates on this error condition. Call your Sun customer service representative.

Message

```
Can't open checklist file: F
```

The file system checklist file *F* (usually `/etc/mstab`) cannot be opened for reading. `fsck` terminates on this error condition. Check access modes of *F*.

Message

```
Can't stat root
```

`fsck`'s request for statistics about the root directory `"/` failed. This should never happen. `fsck` terminates on this error condition. Call your Sun customer service representative.

Message

```
Can't stat F
Can't make sense out of name F
```

`fsck`'s request for statistics about the file system *F* failed. When running manually, it ignores this file system and continues checking the next file system given. Check access modes of *F*.

Message

```
Can't open F
```

`fsck`'s request attempt to open the file system *F* failed. When running manually, it ignores this file system and continues checking the next file system given. Check access modes of *F*.

Message

```
F: (NO WRITE)
```

Either the `-n` flag was specified or `fsck`'s attempt to open the file system *F* for writing failed. When running manually, all the diagnostics are printed out, but no modifications are attempted to fix them.

Message

```
file is not a block or character device; ok
```

You have given `fsck` a regular file name by mistake. Check the type of the file specified.

Possible responses to the `ok` prompt are:

YES

Ignore this error condition.

NO

Ignore this file system and continue checking the next file system given.

Message

```
UNDEFINED OPTIMIZATION IN SUPERBLOCK (SET TO DEFAULT)
```

The superblock optimization parameter is neither `OPT_TIME` nor `OPT_SPACE`.

Possible responses to the `SET TO DEFAULT` prompt are:

YES

Set the superblock to request optimization to minimize running time of the system. (If optimization to minimize disk space utilization is desired, it can be set using `tunefs(8)`.)

NO

Ignore this error condition.

Message

```
IMPOSSIBLE MINFREE=D IN SUPERBLOCK (SET TO DEFAULT)
```

The superblock minimum space percentage is greater than 99% or less than 0%.

Possible responses to the `SET TO DEFAULT` prompt are:

YES

Set the minfree parameter to 10%. (If some other percentage is desired, it can be set using `tunefs(8)`.)

NO

Ignore this error condition.

Message

```
MAGIC NUMBER WRONG
NCG OUT OF RANGE
CPG OUT OF RANGE
NCYL DOES NOT JIVE WITH NCG*CPG
SIZE PREPOSTEROUSLY LARGE
TRASHED VALUES IN SUPER BLOCK
```

followed by the message:

```
F: BAD SUPER BLOCK: B
USE -b OPTION TO FSCK TO SPECIFY LOCATION OF AN ALTERNATE
SUPER-BLOCK TO SUPPLY NEEDED INFORMATION; SEE fsck(8).
```

The superblock has been corrupted. An alternative superblock must be selected from among those listed by `newfs(8)` when the file system was created. For file systems with a block size less than 32K, specifying `-b32` is a good first choice.

Message

```
INTERNAL INCONSISTENCY: M
```

`fsck's` has had an internal panic, whose message is specified as *M*. This should never happen. Call your Sun customer service representative.

Message

```
CAN NOT SEEK: BLK B (CONTINUE)
```

`fsck's` request for moving to a specified block number *B* in the file system failed. This should never happen. Call your Sun customer service representative.

Possible responses to the CONTINUE prompt are:

YES

Attempt to continue to run the file system check. Often, however the problem will persist. This error condition will not allow a complete check of the file system. A second run of `fsck` should be made to recheck this file system. If the block was part of the virtual memory buffer cache, `fsck` will terminate with the message

```
Fatal I/O error
```

NO

Terminate the program.

Message

```
CAN NOT READ: BLK B (CONTINUE)
```

`fsck`'s request for reading a specified block number *B* in the file system failed. This should never happen. Call your Sun customer service representative.

Possible responses to the CONTINUE prompt are:

YES

Attempt to continue to run the file system check. It will retry the read and print out the message:

```
THE FOLLOWING SECTORS COULD NOT BE READ: N
```

where *N* indicates the sectors that could not be read. If `fsck` ever tries to write back one of the blocks on which the read failed it will print the message:

```
WRITING ZERO'ED BLOCK N TO DISK
```

where *N* indicates the sector that was written with zero's. If the disk is experiencing hardware problems, the problem will persist. This error condition will not allow a complete check of the file system. A second run of `fsck` should be made to recheck this file system. If the block was part of the virtual memory buffer cache, `fsck` will terminate with the message

```
Fatal I/O error
```

NO

Terminate the program.

Message

```
CAN NOT WRITE: BLK B (CONTINUE)
```

`fsck`'s request for writing a specified block number *B* in the file system failed. The disk is write-protected; check the write protect lock on the drive. If that is not the problem, call your Sun customer service representative.

Possible responses to the CONTINUE prompt are:

YES

Attempt to continue to run the file system check. The write operation will be retried with the failed blocks indicated by the message:

THE FOLLOWING SECTORS COULD NOT BE WRITTEN: *N*

where *N* indicates the sectors that could not be written. If the disk is experiencing hardware problems, the problem will persist. This error condition will not allow a complete check of the file system. A second run of `fsck` should be made to recheck this file system. If the block was part of the virtual memory buffer cache, `fsck` will terminate with the message

Fatal I/O error

NO

Terminate the program.

Message

bad inode number *DDD* to *ginode*

An internal error has attempted to read non-existent inode *DDD*. This error causes `fsck` to exit. Call your Sun customer service representative.

Phase 1 – Check Blocks and Sizes

This phase concerns itself with the inode list. This section lists error conditions resulting from checking inode types, setting up the zero-link-count table, examining inode block numbers for bad or duplicate blocks, checking inode size, and checking inode format. All errors in this phase except `INCORRECT BLOCK COUNT` and `PARTIALLY TRUNCATED INODE` are fatal if the file system is being preened.

Message

UNKNOWN FILE TYPE *I=I* (CLEAR)

The mode word of the inode *I* indicates that the inode is not a special block inode, special character inode, socket inode, regular inode, symbolic link, FIFO file, or directory inode.

Possible responses to the CLEAR prompt are:

YES

De-allocate inode *I* by zeroing its contents. This will always invoke the `UNALLOCATED` error condition in Phase 2 for each directory entry pointing to this inode.

NO

Ignore this error condition.

Message

PARTIALLY TRUNCATED INODE *I=I* (SALVAGE)

`fsck` has found inode *I* whose size is shorter than the number of blocks allocated to it. This condition should only occur if the system crashes while in the

midst of truncating a file. When preening the file system, `fsck` completes the truncation to the specified size.

Possible responses to the `SALVAGE` prompt are:

YES

Complete the truncation to the size specified in the inode.

NO

Ignore this error condition.

Message

```
LINK COUNT TABLE OVERFLOW (CONTINUE)
```

An internal table for `fsck` containing allocated inodes with a link count of zero cannot allocate more memory. Increase the virtual memory for `fsck`.

Possible responses to the `CONTINUE` prompt are:

YES

Continue with the program. This error condition will not allow a complete check of the file system. A second run of `fsck` should be made to recheck this file system. If another allocated inode with a zero link count is found, this error condition is repeated.

NO

Terminate the program.

Message

```
B BAD I=I
```

Inode *I* contains block number *B* with a number lower than the number of the first data block in the file system or greater than the number of the last block in the file system. This error condition may invoke the `EXCESSIVE BAD BLKS` error condition in Phase 1 (see next message) if inode *I* has too many block numbers outside the file system range. This error condition will always invoke the `BAD/DUP` error condition in Phase 2 and Phase 4.

Message

```
EXCESSIVE BAD BLKS I=I (CONTINUE)
```

There is more than a tolerable number (usually 10) of blocks with a number lower than the number of the first data block in the file system or greater than the number of last block in the file system associated with inode *I*.

Possible responses to the `CONTINUE` prompt are:

YES

Ignore the rest of the blocks in this inode and continue checking with the next inode in the file system. This error condition will not allow a complete check of the file system. A second run of `fsck` should be made to recheck this file system.

NO

Terminate the program.

Message

```
BAD STATE DDD TO BLKERR
```

An internal error has scrambled `fsck`'s state map to have the impossible value `DDD`. `fsck` exits immediately. Call your Sun customer service representative.

Message

```
B DUP I=I
```

Inode `I` contains block number `B` that is already claimed by another inode. This error condition may invoke the `EXCESSIVE DUP BLKS` error condition in Phase 1 if inode `I` has too many block numbers claimed by other inodes. This error condition will always invoke Phase 1b and the `BAD/DUP` error condition in Phase 2 and Phase 4.

Message

```
BAD MODE: MAKE IT A FILE?
```

This error is generated when the status of a given inode is set to all ones, indicating file system damage. This does not indicate disk damage, unless you repeatedly get the message after running `fsck -y`. Respond `y` to the message. `fsck` then reinitializes the inode to a reasonable value.

Message

```
EXCESSIVE DUP BLKS I=I (CONTINUE)
```

There is more than a tolerable number (usually 10) of blocks claimed by other inodes.

Possible responses to the `CONTINUE` prompt are:

YES

Ignore the rest of the blocks in this inode and continue checking with the next inode in the file system. This error condition will not allow a complete check of the file system. A second run of `fsck` should be made to recheck this file system.

NO

Terminate the program.

Message

```
DUP TABLE OVERFLOW (CONTINUE)
```

An internal table in `fsck` containing duplicate block numbers cannot allocate any more space. Increase the amount of virtual memory available to `fsck`.

Possible responses to the `CONTINUE` prompt are:

YES

Continue with the program. This error condition will not allow a complete check of the file system. A second run of `fsck` should be made to recheck this file system. If another duplicate block is found, this error condition will repeat.

NO

Terminate the program.

Message

```
PARTIALLY ALLOCATED INODE I=I (CLEAR)
```

Inode *I* is neither allocated nor unallocated.

Possible responses to the `CLEAR` prompt are:

YES

De-allocate inode *I* by zeroing its contents.

NO

Ignore this error condition.

Message

```
INCORRECT BLOCK COUNT I=I (X should be Y) (CORRECT)
```

The block count for inode *I* is *X* blocks, but should be *Y* blocks. When preening the count is corrected.

Possible responses to the `CORRECT` prompt are:

YES

Replace the block count of inode *I* with *Y*.

NO

Ignore this error condition.

Phase 1B: Rescan for More Dup's

When a duplicate block is found in the file system, the file system is re-scanned to find the inode that previously claimed that block. This section lists the error condition when the duplicate block is found.

Message

```
B DUP I=I
```

Inode *I* contains block number *B* that is already claimed by another inode. This error condition will always invoke the `BAD/DUP` error condition in Phase 2. You can determine which inodes have overlapping blocks by examining this error condition and the `DUP` error condition in Phase 1.

Phase 2 – Check Pathnames

This phase concerns itself with removing directory entries pointing to error conditioned inodes from Phase 1 and Phase 1b. This section lists error conditions resulting from root inode mode and status, directory inode pointers in range, directory entries pointing to bad inodes, and directory integrity checks. All errors in this phase are fatal if the file system is being preened, except for directories not being a multiple of the blocks size and extraneous hard links.

Message

```
ROOT INODE UNALLOCATED (ALLOCATE)
```

The root inode (usually inode number 2) has no allocate mode bits. This should never happen.

Possible responses to the ALLOCATE prompt are:

YES

Allocate inode 2 as the root inode. The files and directories usually found in the root will be recovered in Phase 3 and put into `lost+found`. If the attempt to allocate the root fails, `fsck` will exit with the message

```
CANNOT ALLOCATE ROOT INODE
```

NO

Terminate the program.

Message

```
ROOT INODE NOT DIRECTORY (REALLOCATE)
```

The root inode (usually inode number 2) is not directory inode type.

Possible responses to the REALLOCATE prompt are:

YES

Clear the existing contents of the root inode and reallocate it. The files and directories usually found in the root will be recovered in Phase 3 and put into `lost+found`. If the attempt to allocate the root fails, `fsck` will exit with the message:

```
CANNOT ALLOCATE ROOT INODE
```

NO

`fsck` will then prompt with `FIX`

Possible responses to the `FIX` prompt are:

YES

Replace the root inode's type to be a directory. If the root inode's data blocks are not directory blocks, many error conditions will be produced.

NO

Terminate the program.

Message

```
DUPS/BAD IN ROOT INODE (REALLOCATE)
```

Phase 1 or Phase 1b have found duplicate blocks or bad blocks in the root inode (usually inode number 2) for the file system.

Possible responses to the REALLOCATE prompt are:

YES

Clear the existing contents of the root inode and reallocate it. The files and directories usually found in the root will be recovered in Phase 3 and put into `lost+found`. If the attempt to allocate the root fails, `fsck` will exit with the message:

```
CANNOT ALLOCATE ROOT INODE
```

NO

`fsck` will then prompt with CONTINUE.

Possible responses to the CONTINUE prompt are:

YES

Ignore the DUPS/BAD error condition in the root inode and attempt to continue to run the file system check. If the root inode is not correct, then this may result in many other error conditions.

NO

Terminate the program.

Message

```
NAME TOO LONG F
```

An excessively long path name has been found. This usually indicates loops in the file system name space. This can occur if the super user has made circular links to directories. The offending links must be removed by a Sun customer service representative.

Message

```
I OUT OF RANGE I=I NAME=F (REMOVE)
```

A directory entry *F* has an inode number *I* that is greater than the end of the inode list.

Possible responses to the REMOVE prompt are:

YES

Remove the directory entry *F*.

NO

Ignore this error condition.

Message

```
UNALLOCATED I=I OWNER=O MODE=M SIZE=S MTIME=T type=F (REMOVE)
```

A directory or file entry *F* points to an unallocated inode *I*. The owner *O*, mode *M*, size *S*, modify time *T*, and name *F* are printed.

Possible responses to the REMOVE prompt are:

YES

Remove the directory entry *F*.

NO

Ignore this error condition.

Message

```
DUP/BAD I=I OWNER=O MODE=M SIZE=S MTIME=T type=F (REMOVE)
```

Phase 1 or Phase 1b have found duplicate blocks or bad blocks associated with directory or file entry *F*, inode *I*. The owner *O*, mode *M*, size *S*, modify time *T*, and directory name *F* are printed.

Possible responses to the REMOVE prompt are:

YES

Remove the directory entry *F*.

NO

Ignore this error condition.

Message

```
ZERO LENGTH DIRECTORY I=I OWNER=O MODE=M SIZE=S MTIME=T DIR=F  
(REMOVE)
```

A directory entry *F* has a size *S* that is zero. The owner *O*, mode *M*, size *S*, modify time *T*, and directory name *F* are printed.

Possible responses to the REMOVE prompt are:

YES

Remove the directory entry *F*; this will always invoke the BAD/DUP error condition in Phase 4.

NO

Ignore this error condition.

Message

```
DIRECTORY TOO SHORT I=I OWNER=O MODE=M SIZE=S MTIME=T DIR=F (FIX)
```

A directory *F* has been found whose size *S* is less than the minimum size

directory. The owner *O*, mode *M*, size *S*, modify time *T*, and directory name *F* are printed.

Possible responses to the `FIX` prompt are:

YES

Increase the size of the directory to the minimum directory size.

NO

Ignore this directory.

Message

```
DIRECTORY F LENGTH S NOT MULTIPLE OF B (ADJUST)
```

A directory *F* has been found with size *S* that is not a multiple of the directory block size *B*.

Possible responses to the `ADJUST` prompt are:

YES

Round up the length to the appropriate block size. This error can occur on SunOS file systems prior to Release 4.0. Thus when preening the file system only a warning is printed and the directory is adjusted.

NO

Ignore the error condition.

Message

```
DIRECTORY CORRUPTED I=I OWNER=O MODE=M SIZE=S MTIME=T DIR=F
(SALVAGE)
```

A directory with an inconsistent internal state has been found.

Possible responses to the `FIX` prompt are:

YES

Throw away all entries up to the next directory boundary (usually a 512-byte boundary). This drastic action can throw away up to 42 entries, and should be taken only after other recovery efforts have failed.

NO

Skip up to the next directory boundary and resume reading, but do not modify the directory.

Message

```
BAD INODE NUMBER FOR '.' I=I OWNER=O MODE=M SIZE=S MTIME=T DIR=F
(FIX)
```

A directory *I* has been found whose inode number for '.' does not equal *I*.

Possible responses to the `FIX` prompt are:

YES

Change the inode number for '.' to be equal to *I*.

NO

Leave the inode number for '.' unchanged.

Message

```
MISSING '.' I=I OWNER=O MODE=M SIZE=S MTIME=T DIR=F (FIX)
```

A directory *I* has been found whose first entry is unallocated.

Possible responses to the FIX prompt are:

YES

Build an entry for '.' with inode number equal to *I*.

NO

Leave the directory unchanged.

Message

```
MISSING '.' I=I OWNER=O MODE=M SIZE=S MTIME=T DIR=F
CANNOT FIX, FIRST ENTRY IN DIRECTORY CONTAINS F
```

A directory *I* has been found whose first entry is *F*. *fsck* cannot resolve this problem. The file system should be mounted and the offending entry *F* moved elsewhere. The file system should then be unmounted and *fsck* should be run again.**Message**

```
MISSING '.' I=I OWNER=O MODE=M SIZE=S MTIME=T DIR=F
CANNOT FIX, INSUFFICIENT SPACE TO ADD '.'
```

A directory *I* has been found whose first entry is not '.'. *fsck* cannot resolve this problem as it should never happen. Call your Sun customer service representative.**Message**

```
EXTRA '.' ENTRY I=I OWNER=O MODE=M SIZE=S MTIME=T DIR=F (FIX)
```

A directory *I* has been found that has more than one entry for '.'.

Possible responses to the FIX prompt are:

YES

Remove the extra entry for '.'.

NO

Leave the directory unchanged.

Message

```
BAD INODE NUMBER FOR '....' I=I OWNER=O MODE=M SIZE=S MTIME=T DIR=F
(FIX)
```


A directory *I* has been found whose inode number for '..' does not equal the parent of *I*.

Possible responses to the FIX prompt are:

YES

Change the inode number for '..' to be equal to the parent of *I* ('..' in the root inode points to itself).

NO

Leave the inode number for '..' unchanged.

Message

```
MISSING '..' I=I OWNER=O MODE=M SIZE=S MTIME=T DIR=F (FIX)
```

A directory *I* has been found whose second entry is unallocated.

Possible responses to the FIX prompt are:

YES

Build an entry for '..' with inode number equal to the parent of *I* ('..' in the root inode points to itself).

NO

Leave the directory unchanged.

Message

```
MISSING '..' I=I OWNER=O MODE=M SIZE=S MTIME=T DIR=F
CANNOT FIX, SECOND ENTRY IN DIRECTORY CONTAINS F
```

A directory *I* has been found whose second entry is *F*. `fsck` cannot resolve this problem. The file system should be mounted and the offending entry *F* moved elsewhere. The file system should then be unmounted and `fsck` should be run again.

Message

```
MISSING '..' I=I OWNER=O MODE=M SIZE=S MTIME=T DIR=F
CANNOT FIX, INSUFFICIENT SPACE TO ADD '..'
```

A directory *I* has been found whose second entry is not '..'. `fsck` cannot resolve this problem. The file system should be mounted and the second entry in the directory moved elsewhere. The file system should then be unmounted and `fsck` should be run again.

Message

```
EXTRA '..' ENTRY I=I OWNER=O MODE=M SIZE=S MTIME=T DIR=F (FIX)
```

A directory *I* has been found that has more than one entry for '..'.

Possible responses to the FIX prompt are:

YES

Remove the extra entry for '..'.

NO

Leave the directory unchanged.

Message

```
N IS AN EXTRANEOUS HARD LINK TO A DIRECTORY D (REMOVE)
```

`fsck` has found a hard link, *N*, to a directory, *D*. When preening the extraneous links are ignored.

Possible responses to the REMOVE prompt are:

YES

Delete the extraneous entry, *N*.

NO

Ignore the error condition.

Message

```
BAD INODE S TO DESCEND
```

An internal error has caused an impossible state *S* to be passed to the routine that descends the file system directory structure. `fsck` exits. Call your Sun customer service representative.

Message

```
BAD RETURN STATE S FROM DESCEND
```

An internal error has caused an impossible state *S* to be returned from the routine that descends the file system directory structure. `fsck` exits. Call your Sun customer service representative.

Message

```
BAD STATE S FOR ROOT INODE
```

An internal error has caused an impossible state *S* to be assigned to the root inode. `fsck` exits. Call your Sun customer service representative.

Phase 3 – Check Connectivity

This phase concerns itself with the directory connectivity seen in Phase 2. This section lists error conditions resulting from unreferenced directories, and missing or full `lost+found` directories.

Message

```
UNREF DIR I=I OWNER=O MODE=M SIZE=S MTIME=T (RECONNECT)
```

The directory inode *I* was not connected to a directory entry when the file system was traversed. The owner *O*, mode *M*, size *S*, and modify time *T* of directory

inode *I* are printed. When preening, the directory is reconnected if its size is non-zero; otherwise it is cleared.

Possible responses to the RECONNECT prompt are:

YES

Reconnect directory inode *I* to the file system in the directory for lost files (usually `lost+found`). This may invoke the `lost+found` error condition in Phase 3 if there are problems connecting directory inode *I* to `lost+found`. This may also invoke the `CONNECTED` error condition in Phase 3 if the link was successful.

NO

Ignore this error condition. This will always invoke the `UNREF` error condition in Phase 4.

Message

```
NO lost+found DIRECTORY (CREATE)
```

There is no `lost+found` directory in the root directory of the file system; When preening `fsck` tries to create a `lost+found` directory.

Possible responses to the CREATE prompt are:

YES

Create a `lost+found` directory in the root of the file system. This may raise the message:

```
NO SPACE LEFT IN / (EXPAND)
```

See below for the possible responses. Inability to create a `lost+found` directory generates the message:

```
SORRY. CANNOT CREATE lost+found DIRECTORY
```

and aborts the attempt to linkup the lost inode. This will always invoke the `UNREF` error condition in Phase 4.

NO

Abort the attempt to linkup the lost inode. This will always invoke the `UNREF` error condition in Phase 4.

Message

```
lost+found IS NOT A DIRECTORY (REALLOCATE)
```

The entry for `lost+found` is not a directory.

Possible responses to the REALLOCATE prompt are:

YES

Allocate a directory inode, and change `lost+found` to reference it. The previous inode reference by the `lost+found` name is not cleared. Thus it will either be reclaimed as an UNREF'ed inode or have its link count ADJUST'ed later in this phase. Inability to create a `lost+found` directory generates the message:

```
SORRY. CANNOT CREATE lost+found DIRECTORY
```

and aborts the attempt to linkup the lost inode. This will always invoke the UNREF error condition in Phase 4.

NO

Abort the attempt to linkup the lost inode. This will always invoke the UNREF error condition in Phase 4.

Message

```
NO SPACE LEFT IN /lost+found (EXPAND)
```

There is no space to add another entry to the `lost+found` directory in the root directory of the file system. When preening the `lost+found` directory is expanded.

Possible responses to the EXPAND prompt are:

YES

Expand the `lost+found` directory to make room for the new entry. If the attempted expansion fails `fsck` prints the message:

```
SORRY. NO SPACE IN lost+found DIRECTORY
```

and aborts the attempt to linkup the lost inode. This will always invoke the UNREF error condition in Phase 4. Clean out unnecessary entries in `lost+found`. This error is fatal if the file system is being preened.

NO

Abort the attempt to linkup the lost inode. This will always invoke the UNREF error condition in Phase 4.

Message

```
DIR I=I1 CONNECTED. PARENT WAS I=I2
```

This is an advisory message indicating a directory inode *I1* was successfully connected to the `lost+found` directory. The parent inode *I2* of the directory inode *I1* is replaced by the inode number of the `lost+found` directory.

Message

DIRECTORY *F* LENGTH *S* NOT MULTIPLE OF *B* (ADJUST)

A directory *F* has been found with size *S* that is not a multiple of the directory block size *B* (this can reoccur in Phase 3 if it is not adjusted in Phase 2).

Possible responses to the ADJUST prompt are:

YES

Round up the length to the appropriate block size. This error can occur on SunOS file systems prior to Release 4.0. Thus when preening the file system only a warning is printed and the directory is adjusted.

NO

Ignore the error condition.

Message

BAD INODE *S* TO DESCEND

An internal error has caused an impossible state *S* to be passed to the routine that descends the file system directory structure. `fsck` exits. Call your Sun customer service representative.

Phase 4 – Check Reference Counts

This phase concerns itself with the link count information seen in Phase 2 and Phase 3. This section lists error conditions resulting from unreferenced files, missing or full `lost+found` directory, incorrect link counts for files, directories, symbolic links, or special files, unreferenced files, symbolic links, and directories, and bad or duplicate blocks in files, symbolic links, and directories. All errors in this phase are correctable if the file system is being preened except running out of space in the `lost+found` directory.

Message

UNREF FILE *I*=*I* OWNER=*O* MODE=*M* SIZE=*S* MTIME=*T* (RECONNECT)

Inode *I* was not connected to a directory entry when the file system was traversed. The owner *O*, mode *M*, size *S*, and modify time *T* of inode *I* are printed. When preening the file is cleared if either its size or its link count is zero, otherwise it is reconnected.

Possible responses to the RECONNECT prompt are:

YES

Reconnect inode *I* to the file system in the directory for lost files (usually `lost+found`). This may invoke the `lost+found` error condition in Phase 4 if there are problems connecting inode *I* to `lost+found`.

NO

Ignore this error condition. This will always invoke the CLEAR error condition in Phase 4.

Message

(CLEAR)

The inode mentioned in the immediately previous error condition cannot be reconnected. This cannot occur if the file system is being preened, since lack of space to reconnect files is a fatal error.

Possible responses to the CLEAR prompt are:

YES

De-allocate the inode mentioned in the immediately previous error condition by zeroing its contents.

NO

Ignore this error condition.

Message

```
NO lost+found DIRECTORY (CREATE)
```

There is no lost+found directory in the root directory of the file system; When preening fsck tries to create a lost+found directory.

Possible responses to the CREATE prompt are:

YES

Create a lost+found directory in the root of the file system. This may raise the message:

```
NO SPACE LEFT IN / (EXPAND)
```

See below for the possible responses. Inability to create a lost+found directory generates the message:

```
SORRY. CANNOT CREATE lost+found DIRECTORY
```

and aborts the attempt to linkup the lost inode. This will always invoke the UNREF error condition in Phase 4.

NO

Abort the attempt to linkup the lost inode. This will always invoke the UNREF error condition in Phase 4.

Message

```
lost+found IS NOT A DIRECTORY (REALLOCATE)
```

The entry for lost+found is not a directory.

Possible responses to the REALLOCATE prompt are:

YES

Allocate a directory inode, and change `lost+found` to reference it. The previous inode reference by the `lost+found` name is not cleared. Thus it will either be reclaimed as an UNREF'ed inode or have its link count ADJUST'ed later in this phase. Inability to create a `lost+found` directory generates the message:

```
SORRY. CANNOT CREATE lost+found DIRECTORY
```

and aborts the attempt to linkup the lost inode. This will always invoke the UNREF error condition in Phase 4.

NO

Abort the attempt to linkup the lost inode. This will always invoke the UNREF error condition in Phase 4.

Message

```
NO SPACE LEFT IN /lost+found (EXPAND)
```

There is no space to add another entry to the `lost+found` directory in the root directory of the file system. When preening the `lost+found` directory is expanded.

Possible responses to the EXPAND prompt are:

YES

Expand the `lost+found` directory to make room for the new entry. If the attempted expansion fails `fsck` prints the message:

```
SORRY. NO SPACE IN lost+found DIRECTORY
```

and aborts the attempt to linkup the lost inode. This will always invoke the UNREF error condition in Phase 4. Clean out unnecessary entries in `lost+found`. This error is fatal if the file system is being preened.

NO

Abort the attempt to linkup the lost inode. This will always invoke the UNREF error condition in Phase 4.

Message

```
LINK COUNT type I=I OWNER=O MODE=M SIZE=S MTIME=T COUNT=X  
SHOULD BE Y (ADJUST)
```

The link count for inode *I* is *X* but should be *Y*. The owner *O*, mode *M*, size *S*, and modify time *T* are printed. When preening the link count is adjusted unless the number of references is increasing, a condition that should never occur unless precipitated by a hardware failure. When the number of references is increasing under preen mode, `fsck` exits with the message:

LINK COUNT INCREASING

Possible responses to the ADJUST prompt are:

YES

Replace the link count of file inode *I* with *Y*.

NO

Ignore this error condition.

Message

UNREF *type* I=*I* OWNER=*O* MODE=*M* SIZE=*S* MTIME=*T* (CLEAR)

Inode *I* was not connected to a directory entry when the file system was traversed. The owner *O*, mode *M*, size *S*, and modify time *T* of inode *I* are printed. When preening, this is a file that was not connected because its size or link count was zero, hence it is cleared.

Possible responses to the CLEAR prompt are:

YES

De-allocate inode *I* by zeroing its contents.

NO

Ignore this error condition.

Message

BAD/DUP *type* I=*I* OWNER=*O* MODE=*M* SIZE=*S* MTIME=*T* (CLEAR)

Phase 1 or Phase 1b have found duplicate blocks or bad blocks associated with inode *I*. The owner *O*, mode *M*, size *S*, and modify time *T* of inode *I* are printed. This error cannot arise when the file system is being preened, as it would have caused a fatal error earlier.

Possible responses to the CLEAR prompt are:

YES

De-allocate inode *I* by zeroing its contents.

NO

Ignore this error condition.

Phase 5 – Check Cyl Groups

This phase concerns itself with the free block and used inode maps. This section lists error conditions resulting from allocated blocks in the free block maps, free blocks missing from free block maps, and the total free block count incorrect. It also lists error conditions resulting from free inodes in the used inode maps, allocated inodes missing from used inode maps, and the total used inode count incorrect.

Message

```
CG C: BAD MAGIC NUMBER
```

The magic number of cylinder group *C* is wrong. This usually indicates that the cylinder group maps have been destroyed. When running manually, the cylinder group is marked as needing to be reconstructed. This error is fatal if the file system is being preened.

Message

```
BLK(S) MISSING IN BIT MAPS (SALVAGE)
```

A cylinder group block map is missing some free blocks. During preening the maps are reconstructed.

Possible responses to the `SALVAGE` prompt are:

YES

Reconstruct the free block map.

NO

Ignore this error condition.

Message

```
SUMMARY INFORMATION BAD (SALVAGE)
```

The summary information was found to be incorrect. When preening, the summary information is recomputed.

Possible responses to the `SALVAGE` prompt are:

YES

Reconstruct the summary information.

NO

Ignore this error condition.

Message

```
FREE BLK COUNT(S) WRONG IN SUPERBLOCK (SALVAGE)
```

The superblock free block information was found to be incorrect. When preening, the superblock free block information is recomputed.

Possible responses to the `SALVAGE` prompt are:

YES

Reconstruct the superblock free block information.

NO

Ignore this error condition.

Cleanup

Once a file system has been checked, a few cleanup functions are performed. This section lists advisory messages about the file system and modify status of the file system.

Message

```
V files, W used, X free (Y frags, Z blocks)
```

This is an advisory message indicating that the file system checked contained *V* files using *W* fragment sized blocks, leaving *X* fragment sized blocks free in the file system. The numbers in parenthesis break the free count down into *Y* free fragments and *Z* free full sized blocks.

Message

```
***** REBOOT THE SYSTEM *****
```

This is an advisory message indicating that the root file system has been modified by *fsck*. If the operating system is not rebooted immediately, the work done by *fsck* may be undone by the in-core copies of tables the system keeps. When preening, *fsck* will exit with a code of 4. The auto-reboot script interprets an exit code of 4 by issuing a reboot system call.

Message

```
***** FILE SYSTEM WAS MODIFIED *****
```

This is an advisory message indicating that the current file system was modified by *fsck*. If this file system is mounted or is the current root file system, *fsck* should be halted and the system rebooted. If the system is not rebooted immediately, the work done by *fsck* may be undone by the in-core copies of tables the system keeps.

24.4. References

- [Dolotta78] Dolotta, T. A., and Olsson, S. B. eds., *UNIX User's Manual, Edition 1.1*, January 1978.
- [Joy83] Joy, W., Cooper, E., Fabry, R., Leffler, S., McKusick, M., and Mosher, D. *System Interface Overview*, California University of Computer Systems Research Group Technical Report #4, 1982.
- [McKusick84] McKusick, M., Joy, W., Leffler, S., and Fabry, R. A Fast File System for UNIX, University of California at Berkeley, *ACM Transactions on Computer Systems* 2, 3. pp. 181-197, August 1984.
- [Ritchie78] Ritchie, D. M., and Thompson, K., The UNIX Time-Sharing System, *The Bell System Technical Journal* 57, 6 (July-August 1978, Part 2), pp. 1905-29.

[Thompson78]

Thompson, K., UNIX Implementation, *The Bell System Technical Journal* 57, 6 (July-August 1978, Part 2), pp. 1931-46.

Modifying the `termcap` File

`termcap` is a database describing terminals, used, for example by `vi(1)` and `curses(3X)`. Terminals are described in `termcap` by giving a set of capabilities that they have and by describing how operations are performed. Padding requirements and initialization sequences are included in `termcap`.

Entries in `termcap` consist of a number of colon-separated (`:`) fields. The first entry for each terminal gives the names that are known for the terminal, separated by pipe (`|`) characters. The first name is always two characters long and is used by older systems that store the terminal type in a 16-bit word in a system-wide database. The second name given is the most common abbreviation for the terminal: the last name given should be a long name fully identifying the terminal: and all others are understood as synonyms for the terminal name. All names but the first and last should be in lowercase and contain no blanks; the last name may well contain upper case and blanks for readability.

Terminal names (except for the last, verbose entry) should be chosen using the following conventions. The particular piece of hardware making up the terminal should have a root name chosen, thus “hp2621”. This name should not contain hyphens. Modes that the hardware can be in or user preferences should be indicated by appending a hyphen and an indicator of the mode. Therefore, a “vt100” in 132-column mode would be “vt100-w”. The following suffixes should be used where possible:

Suffix	Meaning	Example
-w	Wide mode (more than 80 columns)	vt100-w
-am	With automatic margins (usually default)	vt100-am
-nam	Without automatic margins	vt100-nam
- <i>n</i>	Number of lines on the screen	aaa-60
-na	No arrow keys (leave them in local)	concept100-na
-np	Number of pages of memory	concept100-4p
-rv	Reverse video	concept100-rv

25.1. Types of Capabilities

Capabilities in `termcap` are of three types: Boolean capabilities, which indicate particular features that the terminal has; numeric capabilities, giving the size of the display or the size of other attributes; and string capabilities, which give character sequences that can be used to perform particular terminal operations. All capabilities have two-letter codes. For instance, the fact that the Concept has

automatic margins , (for example, an automatic return and linefeed when the end of a line is reached) is indicated by the Boolean capability **am**. Hence the description of the Concept includes **am**.

Numeric capabilities are followed by the character # then the value. In the example above **co**, which indicates the number of columns the display has, gives the value 80 for the Concept.

Finally, string-valued capabilities, such as **ce** (clear-to-end-of-line sequence) are given by the two-letter code, an =, then a string ending at the next following colon. A delay in milliseconds may appear after the = in such a capability, which causes padding characters to be supplied by `tputs` after the remainder of the string is sent to provide this delay. The delay can be either a number, *e.g.* 20, or a number followed by an *, for example, 3*. An * indicates that the padding required is proportional to the number of lines affected by the operation, and the amount given is the per-affected-line padding required. (In the case of insert-character, the factor is still the number of *lines* affected; this is always 1 unless the terminal has **in** and the software uses it.) When an * is specified, it is sometimes useful to give a delay of the form 3.5 to specify a delay per line to tenths of milliseconds. (Only one decimal place is allowed.)

A number of escape sequences are provided in the string-valued capabilities for easy encoding of control characters there. **\E** maps to an ESC character, **^X** maps to a control-X for any appropriate X, and the sequences **\n \r \t \b \f** map to linefeed, return, tab, backspace, and formfeed, respectively. Finally, characters may be given as three octal digits after a \, and the characters ^ and \ may be given as ^ and \\. If it is necessary to place a : in a capability it must be escaped in octal as **\072**. If it is necessary to place a NUL character in a string capability it must be encoded as **\200**. (The routines that deal with `termcap` use C strings and strip the high bits of the output very late, so that a **\200** comes out as a **\000** would.)

Sometimes individual capabilities must be commented out. To do this, put a period before the capability name. For example, see the first **cr** and **ta** in the example above.

25.2. Preparing Descriptions

We now outline how to prepare descriptions of terminals. The most effective way to prepare a terminal description is by imitating the description of a similar terminal in `termcap` and to build up a description gradually, using partial descriptions with `vi` to check that they are correct. Be aware that a very unusual terminal may expose deficiencies in the ability of the `termcap` file to describe it or bugs in `vi`. To easily test a new terminal description you can set the environment variable `TERMCAP` to the absolute pathname of a file containing the description you are working on and programs will look there rather than in `/usr/text/lib/termcap`. `TERMCAP` can also be set to the `termcap` entry itself to avoid reading the file when starting up a program.

To get the padding for insert-line right (if the terminal manufacturer did not document it), a severe test is to use `vi` to edit `/etc/passwd` at 9600 baud, delete roughly 16 lines from the middle of the screen, then hit the 'u' key several

times quickly. If the display messes up, more padding is usually needed. A similar test can be used for insert-character.

25.3. Basic Capabilities

The number of columns on each line of the display is given by the **co** numeric capability. If the display is a CRT, then the number of lines on the screen is given by the **li** capability. If the display wraps around to the beginning of the next line when the cursor reaches the right margin, then it should have the **am** capability. If the terminal can clear its screen, the code to do this is given by the **cl** string capability. If the terminal overstrikes (rather than clearing the position when a character is overwritten), it should have the **os** capability. If the terminal is a printing terminal, with no soft copy unit, give it both **hc** and **os**. (**os** applies to storage scope terminals, such as the Tektronix 4010 series, as well as to hard copy and APL terminals.) If there is a code to move the cursor to the left edge of the current row, give this as **cr**. (Normally this will be carriage-return, **^M**.) If there is a code to produce an audible signal (bell, beep, *etc.*), give this as **bl**.

If there is a code (such as backspace) to move the cursor one position to the left, that capability should be given as **le**. Similarly, codes to move to the right, up, and down should be given as **nd**, **up**, and **do**, respectively. These *local cursor motions* should not alter the text they pass over; for example, you would not normally use “**nd=**” unless the terminal has the **os** capability, because the space would erase the character moved over.

A very important point here is that the local cursor motions encoded in `termcap` have undefined behavior at the left and top edges of a CRT display. Programs should never attempt to backspace around the left edge, unless **bw** is given, and never attempt to go up off the top using local cursor motions.

In order to scroll text up, a program goes to the bottom left corner of the screen and sends the **sf** (index) string. To scroll text down, a program goes to the top left corner of the screen and sends the **sr** (reverse index) string. The strings **sf** and **sr** have undefined behavior when not on their respective corners of the screen. Parameterized versions of the scrolling sequences are **SF** and **SR**, which have the same semantics as **sf** and **sr** except that they take one parameter and scroll that many lines. They also have undefined behavior except at the appropriate corner of the screen.

The **am** capability tells whether the cursor sticks at the right edge of the screen when text is output there, but this does not necessarily apply to **nd** from the last column. Leftward local motion is defined from the left edge only when **bw** is given; then an **le** from the left edge will move to the right edge of the previous row. This is useful for drawing a box around the edge of the screen, for example. If the terminal has switch-selectable automatic margins, the `termcap` description usually assumes that this feature is on, for example, **am**. If the terminal has a command that moves to the first column of the next line, that command can be given as **nw** (newline). It is permissible for this to clear the remainder of the current line, so if the terminal has no correctly-working CR and LF it may still be possible to craft a working **nw** out of one or both of them.

These capabilities suffice to describe hardcopy and glass-tty terminals. Thus the Teletype model 33 is described as

```
T3|tty33|33|tty|Teletype model 33:\
:bl=^G:co#72:cr=^M:do=^J:hc:os:
```

and the Lear Siegler ADM-3 is described as

```
l3|adm3|3|LSI ADM-3:\
:am:bl=^G:cl=^Z:co#80:cr=^M:do=^J:le=^H:li#24:sf=^J:
```

25.4. Parameterized Strings

Cursor addressing and other strings requiring parameters are described by a parameterized string capability, with `printf(3S)`-like escapes `%x` in it, while other characters are passed through unchanged. For example, to address the cursor the `cm` capability is given, using two parameters: the row and column to move to. (Rows and columns are numbered from zero and refer to the physical screen visible to the user, not to any unseen memory. If the terminal has memory-relative cursor addressing, that can be indicated by an analogous `CM` capability.)

The `%` encodings have the following meanings:

```
%% output '%'
%d output value as in printf %d
%2 output value as in printf %2d
%3 output value as in printf %3d
%. output value as in printf %c
%+x add x to value, then do %.
%>xy if value > x then add y, no output
%r reverse order of two parameters, no output
%i increment by one, no output
%n exclusive-or all parameters with 0140 (Datamedia 2500)
%B BCD (16*(value/10)) + (value%10), no output
%D Reverse coding (value - 2*(value%16)), no output (Delta Data)
```

Consider the Hewlett-Packard 2645, which, to get to row 3 and column 12, needs to be sent `\E&a12c03Y` padded for 6 milliseconds. Note that the order of the row and column coordinates is reversed here and that the row and column are sent as two-digit integers. Thus its `cm` capability is `cm=6\E&%r%2c%2Y`.

The Microterm ACT-IV needs the current row and column sent simply encoded in binary preceded by a `^T`, `cm=^T%.%..` Terminals that use `%.` need to be able to backspace the cursor (`le`) and to move the cursor up one line on the screen (`up`). This is necessary because it is not always safe to transmit `\n`, `^D`, and `\r`, as the system may change or discard them. (Programs using `termcap` must set terminal modes so that tabs are not expanded, so `\t` is safe to send. This turns out to be essential for the Ann Arbor 4080.)

A final example is the Lear Siegler ADM-3a, which offsets row and column by a blank character, thus `cm=\E=%+ %+`.

Row or column absolute cursor addressing can be given as single parameter capabilities `ch` (horizontal position absolute) and `cv` (vertical position absolute). Sometimes these are shorter than the more general two-parameter sequence (as with the Hewlett-Packard 2645) and can be used in preference to `cm`. If there are

parameterized local motions (*e.g.*, move n positions to the right) these can be given as **DO**, **LE**, **RI**, and **UP** with a single parameter indicating how many positions to move. These are primarily useful if the terminal does not have **cm**, such as the Tektronix 4025.

25.5. Cursor Motions

If the terminal has a fast way to home the cursor (to the very upper left corner of the screen), this can be given as **ho**. Similarly, a fast way of getting to the lower left-hand corner can be given as **ll**; this may involve going up with **up** from the home position, but a program should never do this itself (unless **ll** does), because it can make no assumption about the effect of moving up from the home position. Note that the home position is the same as cursor address (0,0): to the top left corner of the screen, not of memory. (Therefore, the `\EH` sequence on Hewlett-Packard terminals cannot be used for **ho**.)

25.6. Area Clears

If the terminal can clear from the current position to the end of the line, leaving the cursor where it is, this should be given as **ce**. If the terminal can clear from the current position to the end of the display, this should be given as **cd**. **cd** must only be invoked from the first column of a line. (Therefore, it can be simulated by a request to delete a large number of lines, if a true **cd** is not available.)

25.7. Insert/Delete Line

If the terminal can open a new blank line before the line containing the cursor, this should be given as **al**; this must be invoked only from the first position of a line. The cursor must then appear at the left of the newly blank line. If the terminal can delete the line that the cursor is on, this should be given as **dl**; this must only be used from the first position on the line to be deleted. Versions of **al** and **dl** which take a single parameter and insert or delete that many lines can be given as **AL** and **DL**. If the terminal has a settable scrolling region (like the VT100), the command to set this can be described with the **cs** capability, which takes two parameters: the top and bottom lines of the scrolling region. The cursor position is, alas, undefined after using this command. It is possible to get the effect of insert or delete line using this command — the **sc** and **rc** (save and restore cursor) commands are also useful. Inserting lines at the top or bottom of the screen can also be done using **sr** or **sf** on many terminals without a true insert/delete line, and is often faster even on terminals with those features.

If the terminal has the ability to define a window as part of memory which all commands affect, it should be given as the parameterized string **wi**. The four parameters are the starting and ending lines in memory and the starting and ending columns in memory, in that order. (This `terminfo` capability is described for completeness. It is unlikely that any `termcap`-using program will support it.)

If the terminal can retain display memory above the screen, then the **da** capability should be given; if display memory can be retained below, then **db** should be given. These indicate that deleting a line or scrolling may bring non-blank lines up from below or that scrolling back with **sr** may bring down non-blank lines.

25.8. Insert/Delete Character

There are two basic kinds of intelligent terminals with respect to insert/delete character that can be described using `termcap`. The most common insert/delete character operations affect only the characters on the current line and shift characters off the end of the line rigidly. Other terminals, such as the Concept-100 and the Perkin Elmer Owl, make a distinction between typed and untyped blanks on the screen, shifting upon an insert or delete only to an untyped blank on the screen which is either eliminated or expanded to two untyped blanks. You can determine the kind of terminal you have by clearing the screen then typing text separated by cursor motions. Type "abc def" using local cursor motions (not spaces) between the "abc" and the "def". Then position the cursor before the abc and put the terminal in insert mode. If typing characters causes the rest of the line to shift rigidly and characters to fall off the end, then your terminal does not distinguish between blanks and untyped positions. If the "abc" shifts over to the "def" which then move together around the end of the current line and onto the next as you insert, then you have the second type of terminal and should give the capability `in`, which stands for "insert null". While these are two logically separate attributes (one line vs. multi-line insert mode, and special treatment of untyped spaces), we have seen no terminals whose insert mode cannot be described with the single attribute.

`termcap` can describe both terminals that have an insert mode and terminals that send a simple sequence to open a blank position on the current line. Give as `im` the sequence to get into insert mode. Give as `ei` the sequence to leave insert mode. Now give as `ic` any sequence that needs to be sent just before each character to be inserted. Most terminals with a true insert mode will not give `ic`; terminals that use a sequence to open a screen position should give it here. (If your terminal has both, insert mode is usually preferable to `ic`. Do not give both unless the terminal actually requires both to be used in combination.) If post-insert padding is needed, give this as a number of milliseconds in `ip` (a string option). Any other sequence that may need to be sent after insertion of a single character can also be given in `ip`. If your terminal needs to be placed into an 'insert mode' and needs a special code preceding each inserted character, then both `im/ei` and `ic` can be given, and both will be used. The `IC` capability, with one parameter *n*, will repeat the effects of `ic` *n* times.

It is occasionally necessary to move around while in insert mode to delete characters on the same line (*e.g.*, if there is a tab after the insertion position). If your terminal allows motion while in insert mode, you can give the capability `mi` to speed up inserting in this case. Omitting `mi` will affect only speed. Some terminals (notably Datamedia's) must not have `mi` because of the way their insert mode works.

Finally, you can specify `dc` to delete a single character, `DC` with one parameter *n* to delete *n* characters, and delete mode by giving `dm` and `ed` to enter and exit delete mode (which is any mode the terminal needs to be placed in for `dc` to work).

25.9. Highlighting, Underlining, and Visible Bells

If your terminal has one or more kinds of display attributes, these can be represented in a number of different ways. You should choose one display form as *standout mode*, representing a good high-contrast, easy-on-the-eyes format for highlighting error messages and other attention getters. (If you have a choice, reverse video plus half-bright is good, or reverse video alone.) The sequences to enter and exit standout mode are given as `so` and `se`, respectively. If the code to change into or out of standout mode leaves one or even two blank spaces or garbage characters on the screen, as the TVI 912 and Teleray 1061 do, then `sg` should be given to tell how many characters are left.

Codes to begin underlining and end underlining can be given as `us` and `ue`, respectively. Underline mode change garbage is specified by `ug`, similar to `sg`. If the terminal has a code to underline the current character and move the cursor one position to the right, such as the Microterm Mime, this can be given as `uc`.

Other capabilities to enter various highlighting modes include `mb` (blinking), `md` (bold or extra bright), `mh` (dim or half-bright), `mk` (blanking or invisible text), `mp` (protected), `mr` (reverse video), `me` (turn off *all* attribute modes), `as` (enter alternate character set mode), and `ae` (exit alternate character set mode). Turning on any of these modes singly may or may not turn off other modes.

If there is a sequence to set arbitrary combinations of mode, this should be given as `sa` (set attributes), taking 9 parameters. Each parameter is either 0 or 1, as the corresponding attributes is on or off. The 9 parameters are, in order: standout, underline, reverse, blink, dim, bold, blank, protect, and alternate character set. Not all modes need be supported by `sa`, only those for which corresponding attribute commands exist. (It is unlikely that a `termcap`-using program will support this capability, which is defined for compatibility with `terminfo`.)

Terminals with the “magic cookie” glitches (`sg` and `ug`), rather than maintaining extra attribute bits for each character cell, instead deposit special “cookies”, or “garbage characters”, when they receive mode-setting sequences, which affect the display algorithm.

Some terminals, such as the Hewlett-Packard 2621, automatically leave standout mode when they move to a new line or when the cursor is addressed. Programs using standout mode should exit standout mode on such terminals before moving the cursor or sending a newline. On terminals where this is not a problem, the `ms` capability should be present to say that this overhead is unnecessary.

If the terminal has a way of flashing the screen to indicate an error quietly (a bell replacement), this can be given as `vb`; it must not move the cursor.

If the cursor needs to be made more visible than normal when it is not on the bottom line (to change, for example, a non-blinking underline into an easier-to-find block or blinking underline), give this sequence as `vs`. If there is a way to make the cursor completely invisible, give that as `vi`. The capability `ve`, which undoes the effects of both of these modes, should also be given.

If your terminal correctly displays underlined characters (with no special codes needed) even though it does not overstrike, then you should give the capability `ul`. If overstrikes are erasable with a blank, this should be indicated by giving `eo`.

25.10. Keypad

If the terminal has a keypad that transmits codes when the keys are pressed, this information can be given. Note that it is not possible to handle terminals where the keypad only works in local mode (this applies, for example, to the unshifted Hewlett-Packard 2621 keys). If the keypad can be set to transmit or not transmit, give these codes as **ks** and **ke**. Otherwise the keypad is assumed to always transmit. The codes sent by the left-arrow, right-arrow, up-arrow, down-arrow, and home keys can be given as **kl**, **kr**, **ku**, **kd**, and **kh**, respectively. If there are function keys such as **f0**, **f1**, ..., **f9**, the codes they send can be given as **k0**, **k1**, ..., **k9**. If these keys have labels other than the default **f0** through **f9**, the labels can be given as **i0**, **i1**, ..., **i9**. The codes transmitted by certain other special keys can be given: **kH** (home down), **kb** (backspace), **ka** (clear all tabs), **kt** (clear the tab stop in this column), **kC** (clear screen or erase), **kD** (delete character), **kL** (delete line), **kM** (exit insert mode), **kE** (clear to end of line), **kS** (clear to end of screen), **kI** (insert character or enter insert mode), **kA** (insert line), **kN** (next page), **kP** (previous page), **kF** (scroll forward/down), **kR** (scroll backward/up), and **kT** (set a tab stop in this column). In addition, if the keypad has a 3 by 3 array of keys including the four arrow keys, then the other five keys can be given as **K1**, **K2**, **K3**, **K4**, and **K5**. These keys are useful when the effects of a 3 by 3 directional pad are needed. The obsolete **ko** capability formerly used to describe other function keys has been completely supplanted by the above capabilities.

The **ma** entry is also used to indicate arrow keys on terminals that have single-character arrow keys. It is obsolete but still in use in version 2 of *vi* which must be run on some minicomputers due to memory limitations. This field is redundant with **kl**, **kr**, **ku**, **kd**, and **kh**. It consists of groups of two characters. In each group, the first character is what an arrow key sends, and the second character is the corresponding *vi* command. These commands are **h** for **kl**, **j** for **kd**, **k** for **ku**, **l** for **kr**, and **H** for **kh**. For example, the Mime would have `ma=^Hh^Kj^Zk^Xl` indicating arrow keys left (^H), down (^K), up (^Z), and right (^X). (There is no home key on the Mime.)

25.11. Tabs and Initialization

If the terminal needs to be in a special mode when running a program that uses these capabilities, the codes to enter and exit this mode can be given as **ti** and **te**. This arises, for example, from terminals like the Concept with more than one page of memory. If the terminal has only memory-relative cursor addressing and not screen-relative cursor addressing, a screen-sized window must be fixed into the display for cursor addressing to work properly. This is also used for the Tektronix 4025, where **ti** sets the command character to be the one used by `termcap`.

Other capabilities include **is**, an initialization string for the terminal, and **if**, the name of a file containing long initialization strings. These strings are expected to set the terminal into modes consistent with the rest of the `termcap` description. They are normally sent to the terminal by the `tset` program each time the user logs in. They will be printed in the following order: **is**; setting tabs using **ct** and **st**; and finally **if**. (`terminfo` uses **il** before **is** and runs the program **iP** and prints **i3** after the other initializations.) A pair of sequences that does a harder reset from a totally unknown state can be analogously given as **rs** and **if**. These strings are output by the `reset` program, which is used when the terminal gets into

a wedged state. (`terminfo` uses `r1` before `rs` and `r3` after.) Commands are normally placed in `rs` and `rf` only if they produce annoying effects on the screen and are not necessary when logging in. For example, the command to set the VT100 into 80-column mode would normally be part of `is`, but it causes an annoying glitch of the screen and is not normally needed since the terminal is usually already in 80-column mode.

If the terminal has hardware tabs, the command to advance to the next tab stop can be given as `ta` (usually `^I`). A backtab command which moves leftward to the previous tab stop can be given as `bt`. By convention, if the terminal driver modes indicate that tab stops are being expanded by the computer rather than being sent to the terminal, programs should not use `ta` or `bt` even if they are present, since the user may not have the tab stops properly set. If the terminal has hardware tabs that are initially set every n positions when the terminal is powered up, then the numeric parameter it is given, showing the number of positions between tab stops. This is normally used by the `tset` command to determine whether to set the driver mode for hardware tab expansion, and whether to set the tab stops. If the terminal has tab stops that can be saved in nonvolatile memory, the `termcap` description can assume that they are properly set.

If there are commands to set and clear tab stops, they can be given as `ct` (clear all tab stops) and `st` (set a tab stop in the current column of every row). If a more complex sequence is needed to set the tabs than can be described by this, the sequence can be placed in `is` or `if`.

25.12. Delays

Certain capabilities control padding in the terminal driver. These are primarily needed by hardcopy terminals and are used by the `tset` program to set terminal driver modes appropriately. Delays embedded in the capabilities `cr`, `sf`, `le`, `ff`, and `ta` will cause the appropriate delay bits to be set in the terminal driver. If `pb` (padding baud rate) is given, these values can be ignored at baud rates below the value of `pb`. For 4.2BSD `tset`, the delays are given as numeric capabilities `dC`, `dN`, `dB`, `dF`, and `dT` instead.

25.13. Miscellaneous

If the terminal requires other than a NUL (zero) character as a pad, this can be given as `pc`. Only the first character of the `pc` string is used.

If the terminal has commands to save and restore the position of the cursor, give them as `sc` and `rc`.

If the terminal has an extra status line that is not normally used by software, this fact can be indicated. If the status line is viewed as an extra line below the bottom line, then the capability `hs` should be given. Special strings to go to a position in the status line and to return from the status line can be given as `ts` and `fs`. (`fs` must leave the cursor position in the same place that it was before `ts`. If necessary, the `sc` and `rc` strings can be included in `ts` and `fs` to get this effect.) The capability `ts` takes one parameter, which is the column number of the status line to which the cursor is to be moved. If escape sequences and other special commands such as tab work while in the status line, the flag `es` can be given. A string that turns off the status line (or otherwise erases its contents) should be given as `ds`. The status line is normally assumed to be the same width as the rest

of the screen, for example, **co**. If the status line is a different width (possibly because the terminal does not allow an entire line to be loaded), then its width in columns can be indicated with the numeric parameter **ws**.

If the terminal can move up or down half a line, this can be indicated with **hu** (half-line up) and **hd** (half-line down). This is primarily useful for superscripts and subscripts on hardcopy terminals. If a hardcopy terminal can eject to the next page (form feed), give this as **ff** (usually **^L**).

If there is a command to repeat a given character a given number of times (to save time transmitting a large number of identical characters), this can be indicated with the parameterized string **rp**. The first parameter is the character to be repeated and the second is the number of times to repeat it. (This is a `term-
minfo` feature that is unlikely to be supported by a program that uses `termcap`.)

If the terminal has a settable command character, such as the Tektronix 4025, this can be indicated with **CC**. A prototype command character is chosen which is used in all capabilities. This character is given in the **CC** capability to identify it. The following convention is supported on some UNIX systems: The environment is to be searched for a **CC** variable, and if found, all occurrences of the prototype character are replaced by the character in the environment variable. This use of the **cc** environment variable is a very bad idea, as it conflicts with `make(1)`.

Terminal descriptions that do not represent a specific kind of known terminal, such as *switch*, *dialup*, *patch*, and *network*, should include the **gn** (generic) capability so that programs can complain that they do not know how to talk to the terminal. (This capability does not apply to *virtual* terminal descriptions for which the escape sequences are known.)

If the terminal uses `xoff/xon` (DC3/DC1) handshaking for flow control, give **xo**. Padding information should still be included so that routines can make better decisions about costs, but actual pad characters will not be transmitted.

If the terminal has a meta key which acts as a shift key, setting the 8th bit of any character transmitted, then this fact can be indicated with **km**. Otherwise, software will assume that the 8th bit is parity and it will usually be cleared. If strings exist to turn this meta mode on and off, they can be given as **mm** and **mo**.

If the terminal has more lines of memory than will fit on the screen at once, the number of lines of memory can be indicated with **lm**. An explicit value of 0 indicates that the number of lines is not fixed, but that there is still more memory than fits on the screen.

If the terminal is one of those supported by the UNIX system virtual terminal protocol, the terminal number can be given as **vt**.

Media copy strings which control an auxiliary printer connected to the terminal can be given as **ps**: print the contents of the screen; **pf**: turn off the printer; and **po**: turn on the printer. When the printer is on, all text sent to the terminal will be sent to the printer. It is undefined whether the text is also displayed on the terminal screen when the printer is on. A variation **pO** takes one parameter and

leaves the printer on for as many characters as the value of the parameter, then turns the printer off. The parameter should not exceed 255. All text, including **pf**, is transparently passed to the printer while **pO** is in effect.

Strings to program function keys can be given as **pk**, **pl**, and **px**. Each of these strings takes two parameters: the function key number to program (from 0 to 9) and the string to program it with. Function key numbers out of this range may program undefined keys in a terminal-dependent manner. The differences among the capabilities are that **pk** causes pressing the given key to be the same as the user typing the given string; **pl** causes the string to be executed by the terminal in local mode; and **px** causes the string to be transmitted to the computer. Unfortunately, due to lack of a definition for string parameters in `termcap`, only `terminfo` supports these capabilities.

25.14. Glitches and Braindamage

Hazeltine terminals, which do not allow “~” characters to be displayed, should indicate **hz**.

The **nc** capability, now obsolete, formerly indicated Datamedia terminals, which echo `\r\n` for carriage return then ignore a following linefeed.

Terminals that ignore a linefeed immediately after an **am** wrap, such as the Concept, should indicate **xn**.

If **ce** is required to get rid of standout (instead of merely writing normal text on top of it), **xs** should be given.

Teleray terminals, where tabs turn all characters moved over to blanks, should indicate **xt** (destructive tabs). This glitch is also taken to mean that it is not possible to position the cursor on top of a “magic cookie”, and that to erase standout mode it is necessary to use delete and insert line.

The Beehive Superbee, which is unable to correctly transmit the ESC or `^C` characters, has **xb**, indicating that the “f1” key is used for ESC and “f2” for `^C`. (Only certain Superbees have this problem, depending on the ROM.)

Other specific terminal problems may be corrected by adding more capabilities of the form **xx**.

25.15. Similar Terminals

If there are two very similar terminals, one can be defined as being just like the other with certain exceptions. The string capability **tc** can be given with the name of the similar terminal. This capability must be *last*, and the combined length of the entries must not exceed 1024. The capabilities given before **tc** override those in the terminal type invoked by **tc**. A capability can be canceled by placing **xx@** to the left of the **tc** invocation, where *xx* is the capability. For example, the entry

```
hn | 2621-nl:ks@:ke@:tc=2621:
```

defines a `2621-nl` that does not have the **ks** or **ke** capabilities, hence does not turn on the function key labels when in visual mode. This is useful for different modes for a terminal, or for different user preferences.

NOTES:

termcap was replaced by terminfo in UNIX System V Release 2.0. The transition will be relatively painless if capabilities flagged as "obsolete" are avoided.

vi allows only 256 characters for string capabilities, and the routines in term-lib(3) do not check for overflow of this buffer. The total length of a single entry (excluding only escaped newlines) may not exceed 1024.

Not all programs support all entries.

Special Booting Procedures

A.1. Booting from an Alternative Disk

In certain circumstances, you may wish to boot from an alternative device, primarily if the file system on your default booting device becomes corrupted. For disk drives, you can use two alternative ways to boot. If you need to obtain both `/boot` and `/vmunix` from a disk, use the following syntax:

```
> b controller(address, drive, partition)pathname args
```

The arguments in the syntax are explained below.

<i>controller</i>	The disk controller which runs the specific disk: <code>xy</code> for the Xylogics 450 or 451 controller, <code>xd</code> for the Xylogics 7053 controller, or <code>sd</code> for the SCSI disk controller.
<i>address</i>	This is a small number, such as 0 or 1, indicating the <i>n</i> 'th standard controller board, or the physical address of the controller on the bus.
<i>drive</i>	The unit number of the disk on the specified controller.
<i>partition</i>	A number corresponding to the logical partition on the disk where the file specified by <i>pathname</i> can be found. Zero (0) corresponds to Partition a, 1 to Partition b, and so on.
<i>pathname</i>	The name of the file to boot. This can be the name of a file or a <i>symbolic link</i> pointing to a file in the same partition. The symbolic link cannot point to a file actually resident on another disk partition, since only the root partition is mounted by <code>/boot</code> . For example, you can have a link <code>/vmunix</code> pointing to a file <code>/vmunix.new</code> , but not a link <code>/vmunix</code> pointing to <code>/usr/vmunix</code> .
<i>args</i>	Optional arguments that will be passed on to <i>pathname</i> .

You can use the `-a` (for "ask") option of `boot` either to boot from a non-default local file system or a non-default file, as follows:

You can also obtain `/boot` over the network and `/vmunix` from a local disk. If you specify the `-a` option, `boot` itself asks where you wish to boot from. If you do not specify the `-a` option, `/boot` tries to boot `/vmunix` from the same device where it is located, in this case, the network.

```
> b sd()-a
Boot: sd(0,0,0)vmunix -a
root filesystem type (4.2 nfs): 4.2
root device (xy%d[a-h] sd%d[a-h] ): sd0g
root on sd0g fstype 4.2
Boot: vmunix.test

Executable
Size: 380928+87344+69580 bytes
```

In this example you boot `/vmunix.test` from a file system `/dev/sd0g`, (which need have no `/boot`), by using the `/boot` file obtained from `/dev/sd0a`.

An extra complication arises when you obtain `/vmunix` from a non-default file system or a non-default file. Some user programs, notably `ps` and `rpc.rstatd`, assume that they can read the object code of the running kernel from a file called `/vmunix` in the root file system. You can arrange this by creating a symbolic link in the root file system called `/vmunix` that points to the kernel that was really booted.

The following example shows how to boot using a `/boot` file obtained over the network and the `vmunix` file on your machine's local disk.

```
> b ie()-a

Boot: ie(0,0,0)vmunix -a
Using IP Address 192.9.1.90 = C009015A.
Booting from tftp server at 192.9.1.3 = C0090103.
Downloaded 132443 bytes from tftp server.
Boot V1.0
root filesystem type ( 4.2 nfs ): 4.2
root device (xy%d[a-h] sd%d[a-h] ): sd0a
root on sd0a fstype 4.2
Boot: vmunix

Executable
Size: 380928+87344+69580 bytes
```

Here your machine's monitor issues a TFTP request and subsequently obtains `/boot` from an NFS server on the network. The `-a` option causes `/boot` to ask from which device the root file system should be mounted, with 4.2 (disk) and NFS as alternative choices. The `/boot` code then lists the available block devices, `xy`, `xd` and `sd`, then finally mounts `sd0a` as the root file system. It then reads the default program `/vmunix` from this file system, and boots it, passing the `-a` option to it also.

A.2. Booting from an NFS-Mounted Partition

If you have a dataless client machine or standalone system on a network, you may want to boot it from an NFS mounted partition. Note that booting from a local disk does *not* involve the `bootparamd` daemon. However, you can set up a machine with a local disk to boot from an NFS-mounted file system. You do this by placing an entry for the machine in the `bootparams` database `/etc/bootparams` (or `bootparams NIS map`). Specifically, you can obtain `/boot` from a local disk, and subsequently obtain `/vmunix` from an NFS server on the network, as shown below:

```
> b -a

Boot: xy(0,0,0)vmunix -a
root filesystem type ( 4.2 nfs ): nfs
Requesting Internet address for 8:0:20:1:3:36
Internet address is 192.9.1.90
hostname: guppy
domainname: wseng.sun.com
root name: root
server name 'estale'
server_path '/export/root/guppy'
root on estale:/export/root/guppy fstype nfs
Boot: vmunix

Executable
Size: 380928+87344+69580 bytes
```

In this example, the monitor obtains `/boot` from the local `xy` disk. Since it was invoked with the `-a` option, `/boot` does not attempt to locate the default device (the device from which it itself was obtained). Instead, it asks what kind of file system to mount. If the response had been **4.2**, indicating a local file system, `/boot` would have located the default device and mounted the required local file system. Since the response was **nfs**, `/boot` locates a network interface, and completes the boot by mounting an NFS partition from a server. Since `-a` was used, it also asks for the name of the file to boot. The kernel also reacts to the `-a` by asking where to mount its root partition from.

Machines with local disks can also obtain `/boot` and `/vmunix` from an NFS server, even if this is not the default location for these files. To perform this type of boot, use the following syntax:

```
> b controller(address, hostnumber, partition)pathname args
```

The parameters in the command are explained below:

controller

The device abbreviation for the Ethernet controller: `ec` for a 3Com Ethernet controller, `ie` for a Sun-3 or Sun-4 Ethernet controller, or `le` for a LANCE Chip controller.

address

A small number, such as 0 or 1, indicating the *n*'th standard controller board, or the physical address of the controller on the bus.

<i>hostnumber</i>	Should be zero.
<i>partition</i>	Should be zero.
<i>pathname</i>	Pathname of the file to boot.
<i>args</i>	Optional arguments.

A.3. Booting from Tape

You can boot a Sun computer from 1/2 inch nine-track magnetic tape, from a 1/4 inch cartridge tape controlled by a Sun 1/4-inch tape controller, or from a 1/4-inch tape controlled by a SCSI tape controller. Use the following command:

```
> b tape(controller, unit, filenum) -a
```

where

<i>tape</i>	The device abbreviation for the tape controller: <i>mt</i> for 1/2-inch tape with a Tapemaster controller, <i>xt</i> for 1/2-inch tape with a Xylogics controller, <i>ax</i> for a Sun 1/4-inch tape controller, or <i>st</i> for a SCSI tape controller.
<i>controller</i>	A small number, such as 0 or 1, indicating the <i>n</i> th standard magnetic tape controller in the system, or the bus address of the controller.
<i>unit</i>	Specifies which tape drive on the controller is to be used.
<i>filenum</i>	Specifies which file of the tape contains the boot program. (On a SunOS release tape, this is always File #0.) By convention, boot commands number the first file on the tape #0, the second #1, and so on.

A.4. Booting an Alternate Kernel from the Default Device

To boot any file from the default device, enter:

```
> b pathname args
```

In this manner you can boot an alternate version of the kernel by supplying its name as *pathname* in the example above. This is also useful for booting standalone utility programs like `format` after your disk or network is set up.

B

Error Messages from the Monitor and Boot Program

Message

```
Abort at aaaaaa
```

The monitor has aborted execution of the current program because you entered the “abort sequence” (upper left key held while pressing **A**) from the Sun keyboard, or pressed **BREAK** on a serial console. *aaaaaa* is the address of the next instruction. This address is always the same because the kernel calls the monitor. You can continue the program from there by entering the *c* command.

Message

```
Address Error, addr: xxxxxx at aaaaaa
```

The current program has stopped because it made an invalid memory access. *xxxxxx* is the (invalid) address; *aaaaaa* is an address near the instruction which failed (typically two to ten bytes beyond). There is no general way to recover from this error, except to debug the program.

Message

```
ar: cartridge is write protected
```

The current program is trying to write on an Archive tape cartridge, but the “Safe” switch at the top left corner of the cartridge is set to prevent writing on the tape.

Message

```
ar: xxx error
```

The monitor or boot program is trying to boot from an Archive tape, and encountered an unexpected error. The status bytes *xxxx* can be decoded by looking under “Read Status Command” in the *Archive Product Manual*. This error could be caused by incorrect cables, a bad tape, or other problems.

Message

```
ar: drive not responding
```

The monitor is trying to boot from an Archive tape, but can get no response from the tape drive. This can occur if your system contains an Archive controller board but no tape drive, or if the tape drive's cable is loose or disconnected, or if the tape drive's power is not on.

Message

```
ar: invalid state xx
```

This message indicates that the standalone I/O system has a bug in its Archive driver.

Message

```
ar: no cartridge in drive
```

The monitor or boot program is trying to boot from an Archive tape, but there is no cartridge in the tape drive.

Message

```
ar: no drive
```

The monitor or boot program is trying to boot from an Archive tape, but the specified drive does not exist. Typical Archive configurations include only Drive 0.

Message

```
ar: RDST gave Exception, retrying
```

The current program is trying to use the Archive tape drive, and encountered an error. The error is probably caused by hardware. Check the cable(s) that connect the tape drive to the system.

Message

```
ar: triggered at idle xx
```

This message indicates that the standalone I/O system has a bug in its Archive driver.

Message

```
Auto-boot in progress...
```

The monitor has finished its power-on sequence and is looking for a good device to boot the operating system from.

Message

```
Bad device
```

The current program (possibly the boot program) has tried to open a file without a device name (for example, `xy ()`). This could mean that the boot command you typed had no device name.

Message

```
Bad format
```

The boot program is trying to boot from a file which is not in a standard `a.out(5)` format. The boot program can only boot files that are in this format, which is generated by the `ld(1)` command.

Message

```
bn void dd
```

A standalone program (such as the boot program) is trying to read a file from disk or net disk, and the block number it is trying to read is invalid.

Message

```
Boot :
```

The boot program is waiting for you to specify a device and file name to boot from. The boot program accepts the same commands that the monitor would, without the initial `b`. See the section *Booting from Specific Devices* in the chapter *Booting Up and Shutting Down Your System*.

Message

```
Boot: dev(ctrl,unit,part)name options
```

The monitor or boot program is preparing to boot the specified file from the specified device. Either you typed a boot command, or this is an autoboot after power-on. *dev* is the device type; *ctrl*, *unit*, and *part* are the controller, unit-within-controller, and disk partition number. *name* is the name of the file to boot from, if any; *options* are arguments for the booted program, such as `-s`. If you enter a boot command to the monitor, this message will be printed twice; once by the monitor and once by the boot program.

Message

```
boot failed
```

The boot program has tried to boot the device and/or file you specified, but could not. A preceding message should give more details about why.

Message

```

Boot syntax: b [!][dev(ctrlr,unit,part)] name [options]
boot syntax: dev(ctrlr,unit,part)name

```

You have entered an invalid boot command. This message describes the general format of the boot command to remind you. The first form is used by the monitor; the second (without the *b*) is used by the boot program. Don't type the brackets; they indicate optional parts of the command.

Message

```

Bus Error, addr: xxxxxx at aaaaaa

```

The current program has stopped because it tried to make an invalid memory access. The reason for the error is shown before this message. The memory location being accessed was *xxxxxx*, and the instruction that made the access is near location *aaaaaa*. There is no way to recover from this error, in general, except to debug the program.

Message

```

Can't write files yet...Sorry

```

The current program is trying to write to a disk or network disk file through the standalone I/O system. Writing on files (as opposed to writing on devices) is not supported when running standalone (that is, before booting the kernel).

Message

```

Corrupt label
Corrupt label on head h

```

The monitor or boot program is trying to boot from a disk. The first sector of the disk appears to be a label (as it ought to be), but the checksum on the label is wrong. Try again a few times; if the problem recurs, you should probably relabel your disk. Before trying to relabel your disk, make sure that you know what ought to be in the label — writing the wrong label on the disk is likely to destroy some or all files on the disk.

Message

```

count=ddd?

```

A standalone program is trying to write to a device and has specified a block size that is not a multiple of 512. The write proceeds anyway, but may cause incorrect results.

Message

Damage found, *damage*...

As part of the power-on self test procedure, the monitor has found damage in one or more parts of the system. Report this message to your local service representative or Sun Microsystems Field Service. *Damage* is a list of subsystem names, such as “memory” or “timer.”

Message

```
ec%d: Ethernet jammed.
```

After 16 failed transmissions and back offs using the exponential back off algorithm, the packet was dropped.

Message

```
ec%d: can't handle af%d.
```

The interface was handed a message with addresses formatted in an unsuitable address family; the packet was dropped.

Message

```
Exception ee at aaaaaa
```

The current program has stopped because it received an interrupt. The interrupt could have been caused either by hardware or software. *ee* is the hexadecimal address of the interrupt vector used; you can look it up on a Motorola 68010 or 68000 reference card or CPU manual to see what kind of interrupt has occurred. *aaaaaa* is the address of the instruction where the interrupt occurred. If *ee* is 2c, the partition you are trying to boot from is probably missing its boot track.

Message

```
Extra chars in command
```

Your previous *u* command had extra, unrecognized characters on the end.

Message

```
Fcn space
```

The address space being accessed by the monitor's memory reference commands is defined by Function Code number *n*. See the Motorola 68010 or 68020 CPU manual for more information. This message is printed by the *s* command.

Message

```
For phys part p, No label found.
```

The boot program is trying to boot from a non-zero “physical partition” on a

disk, and can't find a label. Physical partitions are used for disk drives, part of which are fixed and part of which are removable.

Message

-> Give the above information to your service-person.

The monitor has found a hardware problem while executing its power-on self test procedure. The preceding messages describe the error in more detail. Report the problem to your local service staff, or to Sun Microsystems Field Service.

Message

```
Giving up...
```

See Waiting for disk to spin up.... The monitor has given up on waiting for the disk to become ready.

Message

```
ie: cannot initialize
```

The monitor or boot program is trying to boot from an Ethernet controller, and something serious has gone wrong with the board. Call your local Sun Microsystems Field Service person.

Message

```
ie%d: Ethernet jammed
```

Network activity has become so intense that sixteen successive transmission attempts failed, causing the 82586 to give up on the current packet. Another possible cause of this message is a noise source somewhere in the network, such as a loose transceiver connection.

Message

```
ie%d: no carrier
```

The 82586 has lost input to its carrier detect pin while trying to transmit a packet, causing the packet to be dropped. Possible causes include an open circuit somewhere in the network and noise on the carrier detect line from the transceiver.

Message

```
ie%d: lost interrupt: resetting
```

The driver and 82586 chip have lost synchronization with each other. The driver recovers by resetting itself and the chip.

Message

```
ie%d: iebark reset
```

The 82586 failed to complete a watchdog timeout command in the allotted time. The driver recovers by resetting itself and the chip.

Message

```
ie%d: WARNING: requeueing
```

The driver has run out of resources while getting a packet ready to transmit. The packet is put back on the output queue for retransmission after more resources become available.

Message

```
ie%d: panic: scb overwritten
```

The driver has discovered that memory that should remain unchanged after initialization has become corrupted. This error usually is a symptom of a bad 82586 chip.

Message

```
ID PROM INVALID
```

The monitor cannot find a valid ID PROM on the CPU board. The ID PROM contains the machine's serial number and other information specific to your system. If you have recently changed CPU boards, it is possible that you installed the ID PROM incorrectly. You should attempt to locate the correct ID PROM and install it in your CPU board.

Message

```
Invalid Page Bus Error ...
```

See Bus Error....

The attempted access was invalid because the virtual page containing the addressed data has been designated as invalid. It usually means that your program is using the wrong address.

Message

```
Invalid selection
```

Your last u command was not correct.

Message

```
Keyboard error detected
```

The microprocessor on the keyboard has reported an error. This probably means

that your keyboard hardware is broken and should be replaced.

Message

Note: The LANCE Ethernet chip is a boot device option only on Sun-3/50s and 3/60s. Other models do not support this chip

```
le%d: transmitter frozen -- resetting
```

A bug in the LANCE chip has caused the chip's transmitter section to stop. The driver has detected this condition and reinitialized the chip.

Message

```
le%d: out of mbufs: output packet dropped
```

The driver has run out of memory to use to buffer packets on output. The packet being transmitted at the time of occurrence is lost. This error is usually symptomatic of trouble elsewhere in the kernel.

Message

```
le%d: stray transmitter interrupt
```

The LANCE chip has signaled that it completed transmitting a packet but the driver has sent no such packet.

Message

```
le%d: LANCE Rev C/D Extra Byte(s) bug; Packet dropped
```

The LANCE chip's internal silo pointers have become misaligned. This error arises from a chip bug.

Message

```
le%d: trailer error
```

An incoming packet claimed to have a trailing header but did not.

Message

```
le%d: runt packet
```

An incoming packet's size was below the Ethernet minimum transmission size.

Message

```
le%d: Receive buffer error - BUFF bit set in rmd
```

This error should never happen, as it occurs only in conjunction with a LANCE feature that the driver does not use.

Message

```
le%d: Received packet with STP bit in rmd cleared
```

This error should never happen, as it occurs only in conjunction with a LANCE feature that the driver does not use.

Message

```
le%d: Received packet with ENP bit in rmd cleared
```

This error should never happen, as it occurs only in conjunction with a LANCE feature that the driver does not use.

Message

```
le%d: Transmit buffer error - BUFF bit set in tmd
```

Excessive bus contention has prevented the LANCE chip from gathering packet contents quickly enough to sustain the packet's transmission over the Ethernet. The affected packet is lost.

Message

```
le%d: Transmit late collision - Net problem?
```

A packet collision has occurred after the channel's slot time has elapsed. This error usually indicates faulty hardware elsewhere on the net.

Message

```
le%d: No carrier - transceiver cable problem?
```

The LANCE chip has lost input to its carrier detect pin while trying to transmit a packet.

Message

```
le%d: Transmit retried more than 16 times - net jammed
```

Network activity has become so intense that sixteen successive transmission attempts failed, causing the LANCE chip to give up on the current packet.

Message

```
le%d: missed packet
```

The driver has dropped an incoming packet because it had no buffer space for it.

Message

```
le%d: Babble error - sent a packet longer than the maximum length
```

While transmitting a packet, the LANCE chip has noticed that the packet's length exceeds the maximum allowed for Ethernet. This error indicates a kernel bug.

Message

```
le%d: Memory Error! Ethernet chip memory access timed out
```

The LANCE chip timed out while trying to acquire the bus for a DVMA transfer.

Message

```
le%d: Reception stopped
```

Because of some other error, the receive section of the LANCE chip shut down and had to be restarted.

Message

```
le%d: Transmission stopped
```

Because of some other error, the transmit section of the LANCE chip shut down and had to be restarted.

Message

```
Load: dev (ctlr,unit,part) boot
```

The monitor has loaded in the *mini* boot program from a disk drive or network disk. The mini boot is now reading in the real boot program from the disk. The real boot program will then read in the program you requested.

Message

```
Lower Byte Parity Bus Error ...
```

See `Bus Error...` and `Parity...`. The preceding access was to a word in memory with a parity error in its lower byte.

Message

```
Misplaced label on head n
```

The monitor or boot program is trying to boot from a disk. It has found a label that seems to identify itself as belonging to a different read/write head from the one where the label is written. See `Corrupt label...` above.

Message

```
mt: controller does not initialize
```

The monitor is trying to boot from nine-track tape, and could not get the tape controller to complete its initialization sequence. This might indicate a possible defect in the controller, or incorrect configuration of the controller board.

Message

```
mt: error 0xxx
```

The monitor is trying to boot from nine-track tape, and encountered an unexpected error. The error number *xx* can be decoded by looking in *Appendix C* of the *Tapemaster Product Specification*, which is supplied with your tape drive.

Message

```
mt: unit not ready
```

The monitor is trying to boot from nine-track tape, but the tape drive is not ready. Check to see that the drive is online.

Message

```
No controller at mbio xxxx
```

The monitor is trying to boot, but it can't find a device controller where you asked it to look. You should try another boot command, or make sure that your controller board is plugged in and has all its jumpers and switches set properly.

Message

```
No default boot devices
```

The monitor is trying to boot but it can't find a disk or Ethernet interface to boot from. To boot from a tape, you must specify the device name explicitly, as in 'b ar ()'.

Message

```
No label found -- attempting boot anyway.
```

The monitor or boot program is trying to boot from a disk, and can't find a valid label on the disk. This is best fixed by booting a copy of *format (8S)* from a different device (for example, network disk or tape) and using the *verify label* and *label* commands. See the warning under *Corrupt label* above. This error might also be caused by missing or bad disk cables.

Message

```
No more file slots
```

The current program is using the standalone I/O library and has opened too many devices or files.

Message

```
not a directory
```

The current program (possibly the boot program) has tried to open a disk or

network disk file with a pathname, but one of the names in the path is not a directory.

Message

```
name not found
```

The boot program has searched for the requested file, but cannot find it. You can retry your boot command, using * instead of *name*, to get a list of the names that exist in that directory.

Message

```
null path
```

The current program (possibly the boot program) has tried to open a file whose name is empty.

Message

```
PageMap aaaaaa [ss]: xxxxxxxx?
```

The monitor is displaying or modifying a page map entry because you entered a *p* command. *aaaaaa* is the virtual memory address whose map entry is being examined. *ss* is the segment map entry that is being used to map this page map entry and page. *xxxxxxx* is the page map entry itself. You can enter a space and type **RETURN** to get back to command mode.

Message

```
Parity Bus Error ...
```

See *Bus Error...* The attempted access was probably valid, but was canceled because the preceding access was to memory with bad parity. (Parity errors are reported on the memory cycle **after** the failing cycle.) If neither "Upper Byte" nor "Lower Byte" is reported, the parity on both bytes was invalid. The access address printed in the Bus Error message is probably not relevant to the parity error.

Message

```
Please clear keyboard to begin
```

The monitor is trying to listen for your typing on the keyboard, but cannot tell which shift keys are down until you release all the locking keys (**Caps Lock** and **Shift Lock**). Once it has seen all the keys released, it can then track the movements of the keys and typing will work.

Message

Please start it, if necessary, -OR- press any key to quit.

See Waiting for disk to spin up....

Message

Possible boot devices:

You have asked for a list of boot devices with the b ? command.

Message

Protection Bus Error ...

See Bus Error.... The attempted access was invalid because your program is not permitted to access the addressed data in this way; for example, writing to that page is disallowed.

Message

ROM Rev *x*, *mm* memory installed

The monitor is identifying its revision level and the system configuration as part of the power-on sequence. *x* is a letter or phrase indicating which particular version of the monitor is installed; *mm* shows how much memory was found during system configuration at power-on. If an even number of megabytes is installed, *mm* is displayed as *nn*MB; otherwise as *nnnn*KB.

Message

Retensing...

The monitor is attempting to boot from an Archive tape. Its first attempt failed, so it is retensing the tape (winding all the tape from one reel to the other), which makes it much more likely to succeed.

Message

sd%d%c: *cmd how (msg) starting blk %d, blk %d (abs blk %d)*

A command such as read or write encountered an error condition (*how*): either it *failed*, the unit was *restored*, or an operation was *retry*'ed. The *msg* is derived from the error number given by the controller, indicating a condition such as "drive not ready" or "sector not found." The *starting blk* is the first sector of the erroneous command, relative to the beginning of the partition involved. The *blk* is the sector in error, again relative to the beginning of the partition involved. The *abs blk* is the absolute block number of the sector in error.

Message

```
Seek not from beginning of file
```

The current program is using the standalone I/O library and has tried to do an unsupported seek operation.

Message

```
SegMap aaaaaa: xx?
```

The monitor is examining or changing the segment map in response to your recent `m` command. You can enter a space and type `RETURN` to get back to command mode.

Message

```
Self Test completed successfully
```

The monitor has completed its power-on self test without finding any hardware problems.

Message

```
Self Test found a problem in something
```

The monitor has completed its power-on self test and found a problem in some subsystem. *something* describes the general location of the error. Further messages give more details; see `Wrote ...` and `Damage found....`

Message

```
Serial #some_number, Ethernet address n:n:n:n:n
```

The monitor is identifying your machine's serial number and hardware Ethernet address as part of the power-on sequence. The hardware Ethernet address is taken from the ID PROM on the CPU board, and is given as a 6-byte hexadecimal value with a colon between each byte. A typical Ethernet address might be `8:0:20:1:1:A3`.

Message

```
Short read
```

The boot program is trying to boot a program from disk or net disk. It has located the program, but encountered an error while reading it into memory.

Message

```
Size: text +data +bss bytes
```

The boot program is loading in the program you requested. *Text*, *data*, and *bss* are the sizes of the three sections of the program; they are printed as each is read

into memory. After finishing display of this message, the boot program begins execution of your program; further messages can come from it instead of from the boot program or monitor.

Message

```
Timeout Bus Error ...
```

See `Bus Error...`. The attempted access was invalid because no device responded at the addressed location. The program was probably trying to access a device or section of memory that does not exist, or that has gotten into a hung state. If this occurs in response to a boot command, the device you are trying to boot from is not installed in your system.

Message

```
tm: no response from ctrlr cc
```

A standalone program (possibly the boot program) is trying to use the Tapemaster nine-track tape drive, and has encountered an error. This could be caused by a bad or missing tape, loose or misplugged cables, incorrect jumpers on the Tapemaster controller board, or hardware errors. *Nn* can be decoded by looking in the Tapemaster *Product Specification*.

Message

```
Unknown device
```

The current program (possibly the boot program) has tried to use a device which is unknown to the standalone I/O system.

Message

```
Upper Byte Parity Bus Error ...
```

See `Bus Error...` and `Parity...`. The preceding access was to a word in memory with a parity error in its upper byte.

Message

```
ui i, uoo, uaabaud, ubbbaud, uuaaaaa, uecho
```

The monitor is describing its console and serial port configuration in response to a `u` command. *i* is the input device (`k` for keyboard, or `a` or `b` for a serial port); *o* is the output device (`s` for screen, or `a` or `b`); *abaud* and *bbaud* are the baud rates on the serial ports; *aaaaaa* is the address of the Zilog 8530 chip that implements the serial ports, and *echo* is `e` if input echoing is enabled (full duplex) or `ne` if disabled (half duplex).

Message

Using RS232 A input.

The monitor did not find the Sun keyboard, so it is taking input from one of the serial ports on the back of the workstation, marked RS232 A. If this is unexpected, make sure that the keyboard is plugged into the correct socket on the workstation. The keyboard must be plugged in before system power is turned on. If you connect a Sun keyboard after this message appears, you can let the monitor know about the keyboard by entering the Abort sequence. (Hold down the upper left key on the Sun keyboard, and press **A**.) The monitor will switch to using the Sun keyboard, since that's where the Abort was typed. Then type **c** to continue whatever program was running when you aborted.

If you don't want to use a Sun keyboard, connect a normal ASCII terminal to the RS232 A connector on the back panel. Configure the terminal for 9600 baud, no parity, one stop bit. Things that you type on the terminal will be displayed on the Sun video screen, if you have one, or on the terminal's screen.

Message

Waiting for disk to spin up...

The monitor is trying to boot from a disk. The disk is not ready, so the monitor is waiting in the hope that the disk is just starting to spin and will become ready soon. If you get this message when the power has been on for a while, your disk cables are probably loose or misconnected.

Message

Watchdog reset!

The current program has stopped executing with a "bus fault." This is explained in detail in the Motorola 68020 and 68030 manuals. It also occurs on machines with the SPARC architecture. The two most common causes are that low memory (interrupt vectors) has been overwritten, or the system stack pointer is pointing to an invalid address, or the kernel stack overflowed. There is a serious problem, possibly in the kernel you are running, more likely it is in the hardware.

Message

What?

You typed a command that the monitor does not recognize. Try again.

Message

Wrote *wdata* at address *addr*, but read *rdata*

The monitor has completed its power-on self test and found a problem in some subsystem. The preceding "Self Test found a problem..." message describes which part of the system was in error. This message gives more details about the

error. *wdata* is the data that was written into part of the system, or which was expected to be there if the system was functioning normally. *addr* is the address where the data was read and/or written. For memory errors, this is a physical memory address; for other errors, the interpretation of this field depends on what subsystem was being tested. *rdata* is the data that was read back from *addr* and was found to be invalid because it was not the same as *wdata*. This information should be written down and reported to your local Field Service organization, or to Sun Microsystems Field Service.

Message

```
xt: no response from ctrl cc
```

A standalone program (possibly the boot program) is trying to use the Tapemaster nine-track tape drive, and has encountered an error. This could be caused by a bad or missing tape, loose or misplugged cables, incorrect jumpers on the Tapemaster controller board, or hardware errors. *nn* can be decoded by looking in the Xylogics Product Specification.

Message

```
xycn: self test error
```

Self test error in the controller; see the Maintenance and Reference Manual.

Message

```
xycn: WARNING: n bit addresses
```

The controller is strapped incorrectly. Sun systems use 20-bit addresses for Multibus based systems and 24-bit addresses for VMEbus based systems. See the subsection on the Xylogics controller in the appropriate Sun *Hardware Installation Manual* for your machine(s) for instructions on how to set the jumpers on the 450.

Message

```
xyn: unable to read bad sector info
```

The bad sector forwarding information for the disk could not be read.

Message

```
xyn and xyn are of same type (n) with different geometries
```

The Xylogics 450/451 does not support mixing the drive types found on these units on a single controller.

Message

```
xyn: initialization failed
```

The drive could not be successfully initialized.

Message

```
xyn: unable to read label
```

The drive geometry/partition table information could not be read.

Message

```
xyn: Corrupt label
```

The geometry/partition label checksum was incorrect.

Message

```
xyn: offline
```

A drive ready status is no longer detected, so the unit has been logically removed from the system. If the drive ready status is restored, the unit will automatically come back online the next time it is accessed.

Message

```
xync: cmd how (msg) blk #n abs blk #n
```

A command such as read or write encountered an error condition (*how*): either it *failed*, the controller was *reset*, the unit was *restored*, or an operation was *retry*'ed. The *msg* is derived from the error number given by the controller, indicating a condition such as "drive not ready," "sector not found", or "disk write protected." The *blk #* is the sector in error relative to the beginning of the partition involved. The *abs blk #* is the absolute block number of the sector in error. Some fields of the error message may be missing since the information is not always available.

Message

```
xy: init error xx
```

The monitor is trying to boot from the Xylogics disk and has encountered an error. The command being executed at the time is defined by the hexadecimal value *xx* (if present); the block number is *bbbb* (if present), and the particular error is encoded as *nn*. The error and command can be decoded by looking in the Xylogics manual.

Message

```
xy: no bad block info
```

The boot program is trying to read from the Xylogics disk, but can't find the information about bad blocks on the disk. It continues, but if the program attempts to read any bad blocks (which have been remapped to elsewhere on the disk), the attempt will fail.

Message

zero length directory

A standalone program (possibly the boot program) is trying to read a file from disk, but one of the directories in the pathname has no files in it. The file system should be checked and fixed by using `fsck(8)`.

Timezones

C.1. TIMEZONE NAME:**TIMEZONE AREA:**

North America:

US/Eastern	Eastern time zone, U.S.A.
US/Central	Central time zone, U.S.A.
US/Mountain	Mountain time zone, U.S.A.
US/Pacific	Pacific time zone, U.S.A.
US/Pacific-New	Pacific time zone, U.S.A., with proposed twiddling of Daylight Savings Time near election time in Presidential election years
US/Yukon	Yukon time zone, U.S.A.
US/East-Indiana	Eastern time zone, U.S.A., no Daylight Savings Time
US/Arizona	Mountain time zone, U.S.A., no Daylight Savings Time
US/Hawaii	Hawaii
Canada/Newfoundland	Newfoundland
Canada/Atlantic	Atlantic time zone, Canada
Canada/Eastern	Eastern time zone, Canada
Canada/Central	Central time zone, Canada
Canada/East-Saskatchewan	Central time zone, Canada, no Daylight Savings Time
Canada/Mountain	Mountain time zone, Canada
Canada/Pacific	Pacific time zone, Canada
Canada/Yukon	Yukon time zone, Canada

Europe:

GB-Eire	Great Britain and Eire
WET	Western European time
Iceland	Iceland
MET	Middle European time (also known as Central European time)
Poland	Poland
EET	Eastern European time

Turkey
W-SU

Turkey
Western Soviet Union

Asia (including Australia and New Zealand):

PRC
Korea
Japan
Singapore
Hongkong
ROC

People's Republic of China
Republic of Korea
Japan
Singapore
Hong Kong
Republic of China

Australia/Tasmania
Australia/Queensland
Australia/North
Australia/West
Australia/South
Australia/Victoria
Australia/NSW

Tasmania, Australia
Queensland, Australia
Northern Territory
Western Australia
South Australia
Victoria, Australia
New South Wales, Australia

NZ

New Zealand

Other (if the locale isn't listed above); none of these have Daylight Savings Time:

GMT	Greenwich Mean time
GMT-1	1 hours west of Greenwich Mean Time
GMT-2	2 hours west of Greenwich Mean Time
GMT-3	3 hours west of Greenwich Mean Time
GMT-4	4 hours west of Greenwich Mean Time
GMT-5	5 hours west of Greenwich Mean Time
GMT-6	6 hours west of Greenwich Mean Time
GMT-7	7 hours west of Greenwich Mean Time
GMT-8	8 hours west of Greenwich Mean Time
GMT-9	9 hours west of Greenwich Mean Time
GMT-10	10 hours west of Greenwich Mean Time
GMT-11	11 hours west of Greenwich Mean Time
GMT-12	12 hours west of Greenwich Mean Time
GMT+13	13 hours east of Greenwich Mean Time
GMT+12	12 hours east of Greenwich Mean Time
GMT+11	11 hours east of Greenwich Mean Time
GMT+10	10 hours east of Greenwich Mean Time
GMT+9	9 hours east of Greenwich Mean Time
GMT+8	8 hours east of Greenwich Mean Time
GMT+7	7 hours east of Greenwich Mean Time
GMT+6	6 hours east of Greenwich Mean Time
GMT+5	5 hours east of Greenwich Mean Time
GMT+4	4 hours east of Greenwich Mean Time
GMT+3	3 hours east of Greenwich Mean Time

GMT+2	2 hours east of Greenwich Mean Time
GMT+1	1 hours east of Greenwich Mean Time

D

The Orange Book

This appendix explains how Release 4.1 of the SunOS operating system fulfills the level C2 computer security requirements listed in the *Orange Book*.[†] According to the *Orange Book*, there are seven levels of computer security, with C2 being the fifth highest. Although the UNIX system was originally designed to be open, making it easy for software developers to share work in progress, most of the technical requirements for C2 security are already provided in the standard UNIX system. However, some things had to be changed.

The remainder of this appendix quotes the requirements given in the *Orange Book* for C2 security level implementations and describes work that was done to satisfy the requirements.

Glossary

The following terms are used in the quotes from the *Orange Book* that appear in the following sections of this appendix:

- ADP Stands for Automated Data Processing. Defined as: “An assembly of computer hardware, firmware, and software configured for the purpose of classifying, sorting, calculating, computing, summarizing, transmitting and receiving, storing, and retrieving data with a minimum of human intervention.”
- Object Defined as: “A passive entity that contains or receives information. Access to an object potentially implies access to the information it contains. Examples of objects are: records, blocks, pages, segments, files, directories, directory trees, and programs, as well as bits, bytes, words, fields, processors, video displays, keyboards, clocks, printers, network nodes, etc.”
- Process Defined as: “A program in execution. It is completely characterized by a single current execution point (represented by the machine state) and address space.”
- Subject Defined as: “An active entity, generally in the form of a person, process, or device that causes information to flow among objects or changes the system state. Technically, a process/domain pair.”
- TCB Stands for Trusted Computing Base. Defined as: “The totality of protection mechanisms within a computer system – including hardware, firmware, and software – the combination of which is responsible for enforcing a security

[†] The *Orange Book* is a common short name for the *Department of Defense Trusted Computer System Evaluation Criteria*, Department of Defense Computer Security Center, Fort George G. Meade, Maryland 20755, DOD 5200.28-STD, December, 1985. The short name stems from book's orange cover.

policy.” The TCB of SunOS includes the hardware, the kernel, all executables used in the normal process of booting the system and bringing it up multiuser, all executables that run with `setuid` to root privileges, and executables normally run by `root`. (Note that the number of utilities normally run by `root` may be severely restricted in secure environments.)

User Defined as: “Any person who interacts directly with a computer system.”

Discretionary Access Control

Discretionary Access Control is defined as: “A means of restricting access to objects based on the identity of subjects and/or groups to which they belong. The controls are discretionary in the sense that a subject with a certain access permission is capable of passing that permission (perhaps indirectly) on to any other subject.”

The discretionary access control requirements stated in the *Orange Book* are the following:

“The TCB shall define and control access between named users and named objects (e.g. files and programs) in the ADP system. The enforcement mechanism (e.g. `self/group/public` controls, access control lists) shall allow users to specify and control sharing of those objects by named individuals, or defined groups of individuals, or by both. The discretionary access control mechanism shall, either by explicit user action or by default, provide that objects are protected from unauthorized access. These access controls shall be capable of including or excluding access to the granularity of a single user. Access permission to an object by users not already possessing access permission shall only be assigned by authorized users.”

What this really means is that a user, at his discretion, can give away access to files and/or groups of files.

The term *authorized user* refers to someone with superuser privileges. The operating system currently enforces this policy by asking for the `root` password when someone becomes superuser.

The access scheme allows permissions to be set for one user, (the file’s owner), one group of users, (the file’s group), and everyone else. To exclude access to the granularity of a single user, the system administrator must define groups containing appropriate users.

Object Reuse

The object reuse requirements stated in the *Orange Book* are the following:

“When a storage object is initially assigned, allocated, or reallocated to a subject from the TCB’s pool of unused storage objects, the TCB shall assure that the object contains no data for which the subject is not authorized.”

The SunOS operating system already guarantees that disk and memory space allocated to a file or process is cleared before it is made available to a user. This satisfies the requirements in this section.

Identification and Authentication

The identification and authentication requirements stated in the *Orange Book* are the following:

“The TCB shall require users to identify themselves to it before beginning to perform any other actions that the TCB is expected to mediate. Furthermore, the TCB shall use a protected mechanism (e.g. passwords) to authenticate the user’s identity. The TCB shall protect authentication data so that it cannot be accessed by any unauthorized user. The TCB shall be able to enforce individual accountability by providing the capability to uniquely identify each individual ADP system user. The TCB shall also provide the capability of associating this identity with all auditable actions taken by that individual.”

As long as each system user is assigned an individual login ID, the `login` and `passwd` commands used by the UNIX system to validate a user’s identity seem to meet most of these requirements. The only problem is the requirement that authentication data cannot be accessed by any unauthorized user. If only a small subset of users is allowed to see the encrypted password, but all users need access to other data in the password file, the encrypted password will have to be maintained in a separate file. Nor can the `getpwent()` routines return an encrypted password. To remedy this problem, password information has been split into two files: `/etc/passwd` and `/etc/security/passwd.adjunct`. `/etc/passwd` contains everything it did before except the encrypted password. `/etc/security/passwd.adjunct` contains the encrypted password and some security information. The latter file is mode 0600 owned by `root`. Instead of an encrypted password, the `/etc/passwd` file contains the string `##user` in the password field. Likewise, the `/etc/group` file contains the string `#$group` to indicate that the encrypted group password is in `/etc/security/group.adjunct`.

This string is what the `getpwent()` routines return unless the process is running as `root`. Also, YP now provides a secure superuser access across the network for the YP `passwd.adjunct` map. All existing software that uses the password file to verify users’ passwords has to be relinked. Any programs that use `getpwent()`, `getpass()`, `crypt()`, or `strcmp()` as the basis for getting and validating passwords will continue to work correctly without source change, but they need to be relinked to pick up new versions of these library routines. Software that reads `/etc/passwd` directly looking for an encrypted password must be modified more extensively.

Auditing

The audit requirements stated in the *Orange Book* are the following:

“The TCB shall be able to create, maintain, and protect from modification or unauthorized access or destruction an audit trail of accesses to objects it protects. The audit data shall be protected by the TCB so that read access to it is limited to those who are authorized for audit data. The TCB shall be able to record the following types of events: use of identification and authentication mechanisms, introduction of objects into a user’s address space (e.g., file open, program initiation), deletion of objects, and actions taken by computer operators and system administrators and/or system security officers. For each recorded event, the audit record shall identify: date and time of the event, user, type of event, and success or failure of the event. For identification/authentication events the origin of

request (e.g., terminal ID) shall be included in the audit record. For events that introduce an object into a user's address space and for object deletion events the audit record shall include the name of the object. The ADP system administrator shall be able to selectively audit the actions of any one or more users based on individual identity.''

In addition to the old UNIX system accounting package, a new auditing package that meets these requirements has been provided. As in previous releases, accounting is configured off by default. Some potential security problems with auditing are listed below.

Auditing Superuser Activities

At the C2 level, one person may perform the roles of operator, system administrator, and system security officer. As a matter of fact, there may be more than one person who knows the superuser password. However, it is impossible to log in as `root` on C2 secure systems; administrators must become superuser by means of the `su` command.

For B1 secure systems, the roles of operator, system administrator, and system security officer (all of which require some superuser privileges) are filled by different people. Some events involving system security would require action by more than one of these people, and no single user would ever be allowed full access to the security features of the system. The three users would be given restricted environments to audit each action taken and restrict the commands they could execute. Many of the commands would be specialized shell scripts set up to perform tasks securely.

Auditing Versus Time and Disk Space

The old accounting mechanism logs an entry for every process at exit time, if accounting is turned on. The `sa` command produced reports from accounting records that could be filtered by `grep` to get records for a specific user. Since the old accounting log files could easily consume 500K bytes per day, accounting was turned off by default standard Sun releases. If all security-related events are logged, log files could consume at least four times as much space, and possibly much more than that.

The new auditing mechanism allows the system administrator to select particular users and events to audit. The system comes with reasonable defaults, which can be changed to suit specific security needs. As with the old accounting system, administrators can use `grep` to select the records they want. It's important to remember that the more events get audited, the slower the system becomes, and the larger the auditing file grows.

Under the old accounting package, if space runs out in the file system holding log files, accounting turns itself off. Under the new security auditing package, the system tries to write to an alternate filesystem when space runs out. At this point, it is the administrator's responsibility to make a tape of the nearly-full audit filesystem, and clear it out so as to create an empty alternate filesystem. If all audit filesystems are full, the system brings itself down, and cannot be rebooted until there is space on some audit filesystem.

What Events Are Audited

Besides the audit trail of commands executed by the operator, system administrator, or system security officer, the following events probably require logging:

- Execution of the following system calls create or complete connections to objects, change the security of objects, terminate or remove objects, or require or grant special privileges:

adjtime()	link()	recvmsg()	shmctl()
bind()	listen()	rename()	shmget()
chmod()	mkdir()	rmdir()	shutdown()
chown()	mknod()	semctl()	socket()
connect()	mount()	semget()	socketpair()
creat()	msgctl()	setdomainname()	swapon()
exec()	msgget()	setgroups()	truncate()
exit()	nfssvc()	sethostname()	unlink()
fchown()	open()	setpriority()	unmount()
flock()	pipe()	setquota()	vfork()
fork()	quota()	setregid()	vhangup()
ftruncate()	quotactl()	setreuid()	
kill()	reboot()	setrlimit()	
killpg()	recvfrom()	settimeofday()	

- If the transfer of information has to be audited as well, the following system calls may also generate audit records:

close()	munmap()	recv()	shmat() †
getdirentries()	ptrace()	semop()	shmdt()
mmap()	read()	send()	write()
msgrcv()	readlink()	sendmsg()	
msgsnd()	readv()	sendto()	

- The lockscreen, login, passwd, and su commands all read passwords and authenticate them using the `crypt()` library routine. These programs, and others that authenticate passwords, produce an audit record. This isn't done by the program itself, but by the authentication daemon `pwdauthd`. It is the responsibility of each secure application to call `pwdauth()` and `grpauth()` as necessary.

System Architecture

The system architecture requirements stated in the *Orange Book* are the following:

“The TCB shall maintain a domain for its own execution that protects it from external interference or tampering (e.g., by the modification of its code or data structures). Resources controlled by the TCB may be a defined subset of the subjects and objects in the ADP system. The TCB shall isolate the resources to be protected so that they are subject to the access control and auditing requirements.”

† Note that the transfer of data through shared memory is not logged by auditing `shmat()` and `shmdt()`. Logging them does, however, generate audit records that place time stamps around the window during which data could be transferred by the process.

This requirement seems to be fully satisfied by the current file protection mechanism. However, permissions on all parts of the TCB should be checked so that:

1. The `/dev/*mem` and `/vmunix` files should be mode 0640, owned by `root`, with group `kmem`.
2. Since the debugger `adb` must not be capable of reading or writing `/dev/*mem`, `/vmunix`, or any other parts of the TCB, `adb` should not be on the list of privileged commands that can be run by the operator, system administrator, or system security officer.
3. Commands that examine memory, such as `ps`, `ipcs`, and `w`, should be setgid to group `kmem`. Note that even though these programs are not setuid `root`, they are still part of the TCB. Note that `kmem`, with group ID of 2, has been added since SunOS 4.0 to the standard Sun release.
4. All files that are part of the TCB must have permissions that deny write access to all users who are not system users or who are not members of system groups. System users are those with login IDs of `root`, `daemon`, `kmem`, `bin`, and `uucp`. System groups are those with group IDs of `wheel`, `daemon`, `kmem`, and `bin`.

System Integrity

The system integrity requirements stated in the *Orange Book* are the following:

“Hardware and/or software features shall be provided that can be used to periodically validate the correct operation of the on-site hardware and firmware elements of the TCB.”

These requirements are satisfied by standard Sun diagnostics.

Format of Audit Records

This appendix describes the format of different types of audit records. If you write programs to examine the audit trail, or if you want to become expert in the use of `audit`, you need to know about audit record format. There are two general categories of audit records: those made for system calls, and those composed of arbitrary text.

E.1. Header and Data Fields

Audit records consist of header fields and data fields. Each record type has a fixed number of header fields, and each field always has the same meaning. The binary format of audit record headers is defined in the include file `<sys/audit.h>`.

System call audit records have a fixed number of data fields, while text audit records have a variable number of data fields. In the case of text audit records, the composition of each field depends on the generating program, and an event's success or failure (and type of failure).

The program `praudit` displays 11 header fields and all the data fields. Audit records as displayed by `praudit` are defined below. Here are the 11 header fields:

1. record type
2. record event
3. time
4. real user ID
5. audit user ID
6. effective user ID
7. real group ID
8. process ID
9. error code
10. return value
11. label

The remainder of this appendix describes the various types of audit records. Unless otherwise stated, the return values on success are the normal return values, and the return values on failure are meaningless. The meanings of error codes are also standard unless otherwise stated.

For system call audit records, the first line is the name of a system call. The next two header fields are the record type and event type. An audit record contains one or more event types, and multiple event types are OR-ed together. The event types may differ for the same record type. The events are based on options to the command and who is running the command. Square brackets enclose valid event types, each separated by commas. Curly braces enclose optional events.

The data fields are listed next. The process group access list always composes the first set of data fields, but these fields are not listed below. The remaining data fields are listed below, however, followed by the type and meaning of each.

E.2. Audit Records for System Calls

This section describes the audit record format for system calls. The one irregularity is that `core` is treated as a system call, even though it isn't.

access

Record Type = access
Event Types = [dr]
Fields = 3
string = current root
string = current working directory
string = file name

adjtime

Record Type = adjtime
Event Types = [p0]
Fields = 2 types/4 values
long = seconds - adjust by
long = microseconds - and adjust by
long = seconds - old adjustment value
long = microseconds - old adjustment value

chmod

Record Type = chmod
Event Types = [da]
Fields = 4
string = current root
string = current working directory
string = file name
int = new mode

chown

Record Type = chown
Event Types = [da]
Fields = 5
string = current root
string = current working directory
string = file name
int = new user ID (-1 if no change)
int = new group ID (-1 if no change)

chroot	Record Type = chroot Event Types = [p0] Fields = 3 string = current root string = current working directory string = new root name
core	Record Type = core Event Types = [dw] Fields = 3 string = current root string = current working directory string = corefile "core" or "more.core"
creat	Record Type = creat Event Types = [dclw] Fields = 3 string = current root string = current working directory string = file name
execv	Record Type = execv Event Types = [dr] Fields = 3 string = current root string = current working directory string = path name
execve	Record Type = execve Event Types = [dr] Fields = 3 string = current root string = current working directory string = path name
fchmod	Record Type = fchmod Event Types = [da] Fields = 2 int = file descriptor int = new mode

fchown Record Type = chown
Event Types = [da]
Fields = 3
int = file descriptor
int = new user ID (-1 if no change)
int = new group ID (-1 if no change)

ftruncate Record Type = ftruncate
Event Types = [dw]
Fields = 2
int = file descriptor
long = length to truncate to

kill Record Type = kill
Event Types = [dw]
Fields = 2
int = process ID
int = signal number

killpg Record Type = killpg
Event Types = [dw]
Fields = 2
int = process group ID
int = signal number

link Record Type = link
Event Types = [dc]
Fields = 4
string = current root
string = current working directory
string = name of file to link to
string = hard link name

mkdir Record Type = mkdir
Event Types = [dc]
Fields = 3
string = current root
string = current working directory
string = directory name

mknod	Record Type = mknod Event Types = [dc, p0(only if regular user and not a FIFO)] Fields = 4 string = current root string = current working directory string = file name string = mode
mount	Record Type = mount Event Types = [p1] Fields = 6 string = current root string = current working directory int = mount type0: MOUNT_UFS - defined in mount . h string = local mount directory int = flags - file system read/write and suid, etc - defined in mount . h int = mount type specific arguments
mount ufs	Record Type = mount_ufs Event Types = [p1] Fields = 6 string = current root string = current working directory int = mount type0: MOUNT_UFS - defined in mount . h string = local mount directory int = flags - file system read/write and suid, etc - defined in mount . h string = block special file to mount type specific argument

mount nfs

Record Type = mount_nfs
Event Types = [p1]
Fields = 13 types/26 values
string = current root
string = current working directory
int = mount type(1):MOUNT_NFS - defined in mount.h
string = local mount directory name
int = flags - file system attributes - defined in mount.h
sockaddr_in - file server address consists of the following fields:
 short = family AF_INET=2; defined in socket.h
 unsigned short = port number
 hex = IP address may consist of 1 long, 2 shorts or 4 chars
 char = 8 chars
fhandle_t - file handle to be mounted (differs between client and server)
 client:
 NFS_FHSIXE (32) chars = clients view of fhandle
 (opaque data displayed as four hexadecimal values)
 nfs server:
 2 ints = file system ID
 unsigned short = file number
 char = only used to pad structure
int = flags
int = write size in bytes
int = read size in bytes
int = initial timeout in .1 seconds
int = times to retry send
string = remote host name

msgctl

Record Type = msgctl
Event Types = [dc(IPC_RMID), dw(IPC_SET), dr(IPC_STAT)]
Fields = 1
int = message queue ID

msgget

Record Type = msgget
Event Types = [drl dw] { ldc if a new message queue is created }
Fields = 2
int = message queue ID or -1 on error
int = key - type of message queue as defined in ipc.h

msgrcv

Record Type = msgrcv
Event Types = [dr]
Fields = 1
int = message queue ID

msgsnd	Record Type = msgsnd Event Types = [dw] Fields = 1 int = message queue ID
open	Record Type = open Event Types = [dwldr, dw, dr] {ldc} Fields = 3 string = current root string = current working directory string = file name
ptrace	Record Type = ptrace Event Types = [dw] Fields = 5 int = request as defined in ptrace.h int = process ID int = depends on type of request - see ptrace man page int = depends on type of request - see ptrace man page int = depends on type of request - see ptrace man page
quotactl	This is broken down as follows:
quota on	Record Type = quota_on Event Types = [p0] Fields = 6 string = current root string = current working directory int = quota command - as defined in quota.h string = device - character or block int = uid string = quota file name
quota off	Record Type = quota_off Event Types = [p0] Fields = 5 string = current root string = current working directory int = quota command - as defined in quota.h string = device - character or block int = uid

quota set

Record Type = quota_set
Event Types = [p0]
Fields = 6 types/13 values
string = current root
string = current working directory
int = quota command - as defined in quota.h
string = device - character or block
int = uid
dqblk = defines the format of the disk quota file.
unsigned int = absolute limit on disk blks alloc
unsigned int = preferred limit on disk blks
unsigned int = current block count
unsigned int = maximum # allocated files + 1
unsigned int = preferred file limit
unsigned int = current # allocated files
unsigned int = time limit for excessive disk use
unsigned int = time limit for excessive files

quota lim

Record Type = quota_lim
Event Types = [p0]
Fields = 6 types/13 values
string = current root
string = current working directory
int = quota command - as defined in quota.h
string = device - character or block
int = uid
dqblk = defines the format of the disk quota file.
unsigned int = absolute limit on disk blks alloc
unsigned int = preferred limit on disk blks
unsigned int = current block count
unsigned int = maximum # allocated files + 1
unsigned int = preferred file limit
unsigned int = current # allocated files
unsigned int = time limit for excessive disk use
unsigned int = time limit for excessive files

quota sync

Record Type = quota_sync
Event Types = [p0]
Fields = 5
string = current root
string = current working directory
int = quota command - as defined in quota.h
string = device - character or block
int = uid

quota	<p>Record Type = quota Event Types = [p0] Fields = 6 string = current root string = current working directory int = quota command - as defined in quota.h string = device - character or block int = uid string = argument supplied to quota</p>
readlink	<p>Record Type = readlink Event Types = [dr] Fields = 5 string = current root string = current working directory string = link name string = buffer which contain contents of link int = count of characters in buffer</p>
reboot	<p>Record Type = reboot - always generated before reboot call Event Types = [p1] Return Value = successful - AU_EITHER(-1) defined in audit.h Fields = 1 int = argument to reboot - see reboot2 man page</p>
rebootfailure	<p>Record Type = rebootfail - only generated if reboot fails Event Types = [p1] Fields = 1 int = argument to reboot - see reboot2 man page</p>
rename	<p>Record Type = rename Event Types = [dc] Fields = 4 string = current root string = current working directory string = rename from file name string = rename to file name</p>
rmdir	<p>Record Type = rmdir Event Types = [dc] Fields = 3 string = current root string = current working directory string = directory name</p>

semctl	<p>The possible commands are:</p> <p>GETVAL SETVAL GETPID GETNCNT GETZCNT GETALL SETALL IPC_STAT IPC_SET IPC_RMID</p> <p>The event type value for all commands except the one listed below is zero (0).</p> <p>Record Type = semctl Event Types = [dc(IPC_RMID), dw(SETALL), dr(SETZCNT)] Fields = 1 int = semaphore ID</p>
semget	<p>Record Type = semget Event Types = [drldw] { ldc if a new semaphore is created } Fields = 2 int = semaphore ID int = key</p>
semop	<p>Record Type = semop Event Types = [drldw] Fields = 1 int = semaphore ID</p>
setdomainname	<p>Record Type = setdomainname Event Types = [p1] Fields = 2 string = old domain name string = new domain name</p>
sethostname	<p>Record Type = sethostname Event Types = [p1] Fields = 2 string = old host name string = new host name</p>
settimeofday	<p>Record Type = settimeofday Event Types = [p0] Fields = 2 types/4 values long = seconds long = microseconds int = minutes west of Greenwich int = type of dst correction</p>

shmat	<p>Record Type = shmat Event Types = [drldw(if attach is not read only) dc(if first attach, create)] Fields = 1 int = shared memory ID</p>
shmctl	<p>The three shmctl commands are: IPC_STAT IPC_SET IPC_RMID</p> <p>The event type for IPC_SET is zero(0).</p> <p>Record Type = shmctl Event Types = [dc(IPC_RMID, dr(IPC_STAT))] Fields = 1 int = shared memory ID</p>
shmdt	<p>Record Type = shmdt Event Types = [dc] Fields = 1 int = shared memory ID</p>
shmget	<p>Record Type = shmget Event Types = [drldw] {l dc - if creating memory segment} Fields = 2 int = shared memory segment ID or EINVAL if error int = key</p>
socket	<p>Record Type = socket Event Types = [dwldrldc] Fields = 3 int = domain address family; defined in socket.h int = socket type as defined in socket.h int = protocols - see socket(2), services(5) and protocols(5) man pages</p>
stat	<p>Record Type = stat Event Types = [dr] Fields = 3 string = current root string = current working directory string = file name</p>

statfs Record Type = statfs
Event Types = [dr]
Fields = 4 types/19 values
string = current root
string = current working directory
string = directory name fs is mounted on
statfs = file system statistics - consists of the following fields:
long = type of info, zero for now
long = fundamental file system block size
long = total blocks in file system
long = free block in fs
long = free blocks avail to non-superuser
long = total file nodes in file system
long = free file nodes in fs
long = first part of file system ID
long = second part of file system ID
7 longs = spare for later

symlink Record Type = symlink
Event Types = [dc]
Fields = 4
string = current root
string = current working directory
string = file name to link to
string = link name

sysacct Record Type = sysacct
Event Types = [p1|dw]
Fields = 1
string = accounting file name

truncate Record Type = truncate
Event Types = [dw]
Fields = 4
string = current root
string = current working directory
string = file name
long = length to truncate to

unlink Record Type = unlink
Event Types = [dc]
Fields = 3
string = current root
string = current working directory
string = file name

unmount Record Type = unmount
 Event Types = [p1]
 Fields = 3
 string = current root
 string = current working directory
 string = local mount directory

utimes Record Type = utimes
 Event Types = [dw]
 Fields = 4 types/7 values
 string = current root
 string = current working directory
 string = file name to set times on
 times consists of:
 long = time of last access in seconds
 long = and microseconds
 long = time of last modification in seconds
 long = and microseconds

E.3. Audit Records for Arbitrary Text

This section describes the audit record format for arbitrary text. Many programs write text audit records because they perform tasks that don't use system calls, but nevertheless have an impact on security.

text Record Type = text
 Event Types = depends on program
 Fields = number depends on program
 string = content depends on program

In addition to kernel generated audit messages, several programs generate text audit records. The event type, return values, error codes, and data fields depend on the program that generates the audit record. These fields are described below. The data field lines are preceded by the data field position number. The first data field is the name of the command.

audit - audit trail maintenance Event Types = [ad]

1-string = audit
2-string = Invalid input
2-string = Successful

COMMAND = audit -d usernames
2-string = Invalid user name
2-string = Error in getting event state.
2-string = {system error message}
3-string = -d
4-number of users-string = usernames
" " " "
{maximum of ten user names}
" " " "

COMMAND = audit -u username
2-string = Invalid user name
2-string = Error in converting audit flags.
2-string = {system error message}

3-string = -u
4-string = username

COMMAND = audit -s
2-string = Error in audit_data open.
2-string = {system error message}
3-string = -s

COMMAND = audit -n
2-string = Error in audit_data open.
2-string = {system error message}
3-string = -n

clri Event Types = [ad]
1-string = clri
additional strings = parameters to clri

dcheck Event Types = [ad]
1-string = dcheck
additional strings = parameters to dcheck

dump Event Types = [ad]
 1-string = dump
 additional strings = parameters to dump

fsck Event Types = [ad]
 1-string = fsck
 additional strings = parameters to fsck

icheck Event Types = [ad]
 1-string = icheck
 additional strings = parameters to icheck

login - sign on Event Types = [lo]

Return Value = successful	0 {user authenticated}
Return Value = incorrect password	1
Return Value = logins disabled and uid != 0	2
Return Value = ROOT LOGIN REFUSED	3
Return Value = No shell	5

Error Code = same as return values

1-string = login
 2-string = login name
 3-string = message text, as return value
 4-string = terminal name
 5-string = host name

ncheck Event Types = [ad]
 1-string = ncheck
 additional strings = parameters to ncheck

pwdauthd - server for authenticating passwords

Event Types = [ad]

1-string = pwdauthd

2-string = "user" if authentication was for a user

2-string = "group" if authentication was for a group

3-string = remote user's uid

4-string = name of user or group to authenticate

5-string = password to authenticate

6-string = not using passwd.adjunct

6-string = not using group.adjunct

6-string = valid

6-string = invalid

restore

Event Types = [ad]

1-string = restore

additional strings = parameters to restore

rexid - remote execution daemon

Event Types = [lo]

1-string = in.rexd

2-string = command

3-string = remote machine name

4-string =

5-string = rexd: service rpc register: error

5-string = rexd: svctcp_create: error

5-string = rexd: service rpc register: error

5-string = user authorized

rexecd - remote execution daemon

Event Types = [lo]

1-string = in.rexecd
2-string = user name
3-string = remote host name
4-string = command to execute

5-string = Login incorrect
5-string = Password incorrect
5-string = No remote directory

5-string = user authorized

rshd - remote shell daemon

Event Types = [ad]

1-string = in.rshd
2-string = remote user name
3-string = local user name
4-string = host name
5-string = command to execute

6-string = Login incorrect.
6-string = Permission denied
6-string = Can't make pipe
6-string = Error in fork

6-string = authorization successful

**su - super-user, temporarily
switch effective user ID**

Event Types = [ad]

Return Value = successful	0 {user authenticated}
Return Value = Unknown login:	1
Return Value = bad password	2
Return Value = setgid	3
Return Value = initgroups failed	4
Return Value = setuid	5
Return Value = no directory	6
Return Value = no shell	7

Error Code = same as return values

1-string = su
 2-string = message text
 3-string = user name
 4-string = user environment; user
 5-string = home directory
 6-string = shell
 7-string = path

**yppasswdd - server for
modifying NIS password file**

Event Types = [ad]

1-string = yppasswdd
 2-string = remote user name
 3-string = Not in passwd.adjunct.
 3-string = Bad password in passwd.adjunct.
 3-string = Inconsistency between passwd files.
 3-string = Successful.

**passwd - Change user
password, full name, or shell**

Event Types = [ad]

Return Value = Password changed	0
Return Value = Full name changed	0
Return Value = Shell changed	0
Return Value = Program error	1
Return Value = Password file busy	1
Return Value = User not found in passwd file	1
Return Value = Permission denied	1

Error Code = same as return values

1-string = username
 1-string = "", when user name is not known

IP Number Registration Form

You can fill out the following questionnaire and send it to SRI-NIC. It is identical to the one provided by the Hostmaster at SRI-NIC at the time of this printing. When you complete it, send it via electronic mail to `HOSTMASTER@NIC.DDN.MIL`, or, if electronic mail is not available to you, send it by regular mail to:

DDN Network Information Center
 SRI International
 Room EJ217
 333 Ravenswood Avenue
 Menlo Park, CA 94025

Note: If the network will NOT be connected to either the DARPA Internet or the DDN Internet, then you do not need to provide this information.

- 1) If the network will be connected to the DARPA Internet or the DDN Internet, you must provide the name of the sponsoring organization, and the name, title, mailing address, phone number, net mailbox, and NIC Handle (if any) of the contact person (POC) at that organization who has authorized the network connection. This person will serve as the POC for administrative and policy questions about authorization to be a part of the DARPA Internet or the DDN Internet. Examples of such sponsoring organizations are: Defense Communications Agency (DCA), Defense Advanced Research Projects Agency (DARPA), the National Science Foundation (NSF), or similar military or government sponsors.

Example:

Sponsor

Organization	DARPA
Name	Lastname, Firstname
Title	Program Manager
Mail Address	DARPA/ISTO Office 1400 Wilson Boulevard Arlington, VA 22209
Phone Number	(XXX) XXX-XXX
Net Mailbox	progmgr@VAX.DARPA.MIL
NIC Handle	AA12

- 2) Provide the name, title, mailing address, phone number, and organization of the administrative POC for the network requesting the number. This is the

POC for administrative and policy questions about the network itself. If the network is associated with a research project this POC should be the Principal Investigator of the project. The online mailbox and NIC Handle (if any) of this person should also be included.

Example:

Administrator

```
Organization  SRI International
              Network Information Center
Name          Lastname, Firstname
Title        Principal Investigator
Mail Address  SRI International
              333 Ravenswood Avenue
              Menlo Park, CA 94025
Phone Number  (XXX) XXX-XXXX
Net Mailbox   pi@NIC.DDN.MIL
NIC Handle    BB34
```

- 3) Provide the name, title, mailing address, phone number, and organization of the technical POC. The online mailbox and NIC Handle (if any) of the technical POC should also be included. This is the POC for resolving technical problems associated with the network and for updating information about the network. The technical POC may also be responsible for hosts attached to this network.

Example:

Technical POC

```
Organization  SRI International
              Network Information Center
Name          Lastname, Firstname
Title        Computer Scientist
Mail Address  SRI International
              333 Ravenswood Avenue
              Menlo Park, CA 94025
Phone Number  (XXX)XXX-XXXX
Net Mailbox   cs@NIC.DDN.MIL
NIC Handle    CC56
```

- 4) Supply the short mnemonic name for the network (up to 12 characters). This is the name that will be used as an identifier in internet name and address tables.

Example:

ALPHA-BETA

- 5) Supply the descriptive name of the network (up to 20 characters). This name might be used to clarify the ownership, location, or purpose of the network.

Example:

Greek Alphabet Net

- 6) Identify the network geographic location.

Example:

SRI International
Network Information Center
333 Ravenswood Avenue
Menlo Park, CA 94025

- 7) Provide a citation to a document that describes the technical aspects of the network. If the document is online, give a pathname suitable for online retrieval.

Example:

"The Ethernet, a Local Area Network: Data Link Layer and Physical Layer Specification", X3T51/80-50 Xerox, Stamford Connecticut, October 1980.

Note: For networks connected to the DARPA Internet or the DDN Internet the gateway must be either a core gateway supplied and operated by BBN, or a gateway of another Autonomous System. If this gateway is not a core gateway, then an identifiable gateway in this gateway's Autonomous System must exchange routing information with a known core gateway via EGP.

- 8) Gateway information required:

If the network is to be connected to the DARPA Internet or the DDN Internet, answer questions 8a and 8b.

- 8a) Describe the Gateway that connects the new network to the DARPA Internet or the DDN Internet, and the date it will be operational. The gateway must be either a core gateway supplied and operated by BBN, or a gateway of another Autonomous System. If this gateway is not a core gateway, then an identifiable gateway in this gateway's Autonomous System must exchange routing information with a known core gateway via EGP.

A good way to answer this question is to say "Our gateway is supplied by person or company X and does whatever their standard issue gateway does".

Example:

Our gateway is the standard issue supplied and operated by BBN, and will be installed and made operational on 1-April-83.

- 8b)

Describe the gateway machine, including:

- (a) Hardware (LSI-11/23, VAX-11/750, etc. interfaces)
- (b) Addresses (what host on what net for each connected net)
- (c) Software (operating system and programming language)

Example:

- (a) Hardware

PDP-11/40, ARPANET Interface by ACC, Ethernet Interfaces by 3COM.

(b) Address

10.9.0.193 on ARPANET

(c) Software

Berkeley Unix 4.2 BSD and C

9) Estimate the number of hosts that will be on the network:

(a) Initially,

(b) Within one year,

(c) Within two years

(d) Within five years.

Example:

(a) initially = 5
(b) one year = 25
(c) two years = 50
(d) five years = 200

10)

Unless a strong and convincing reason is presented, the network (if it qualifies at all) will be assigned a class C network number. If a class C network number is not acceptable for your purposes state why. (Note: If there are plans for more than a few local networks, and more than 100 hosts, you are strongly urged to consider subnetting. [See RFC 950])

Example:

Class C is fine.

11)

Networks are characterized as being either Research, Defense, Government - Non Defense, or Commercial, and the network address space is shared between these three areas. Which type is this network?

Example:

Research

12)

What is the purpose of the network?

Example:

To economically connect computers used in DARPA sponsored research project FROB-BRAF to the DARPA Internet or the DDN Internet to provide communication capability with other similar projects at UNIV-X and CORP-Y.

PLEASE ALLOW AT LEAST 10 WORKING DAYS FOR PROCESSING THIS APPLICATION

For further information contact the DDN/ARPANET Network Information Center (NIC):

Via electronic mail: HOSTMASTER@NIC.DDN.MIL
 Via telephone: (800) 235-3155
 Via postal mail: SRI International
 DDN Network Information Center
 333 Ravenswood Avenue
 EJ217
 Menlo Park, CA 94025

RECOMMENDED READING (available from the NIC)

Braden, R.T.; Postel, J.B. Requirements for Internet gateways. Marina del Rey, CA: University of Southern California, Information Sciences Inst.; 1987 June; RFC 1009. 55 p. (NIC.DDN.MIL RFC:RFC1009.TXT).

Mogul, J.; Postel, J.B. Internet standard subnetting procedure. Stanford, CA: Stanford University; 1985 August; RFC 950. 18 p. (NIC.DDN.MIL RFC:RFC950.TXT).

Postel, J.B. Internet Control Message Protocol. Marina del Rey, CA: University of Southern California, Information Sciences Inst.; 1981 September; RFC 792. 21 p. (NIC.DDN.MIL RFC:RFC792.TXT).

Postel, J.B. Transmission Control Protocol. Marina del Rey, CA: University of Southern California, Information Sciences Inst.; 1981 September; RFC 793. 85 p. (NIC.DDN.MIL RFC:RFC793.TXT).

Postel, J.B. Address mappings. Marina del Rey, CA: University of Southern California, Information Sciences Inst.; 1981 September; RFC 796. 7 p. (NIC.DDN.MIL RFC:RFC796.TXT).

Obsoletes: IEN 115 (NACC 0968-79)

Postel, J.B. User Datagram Protocol. Marina del Rey, CA: University of Southern California, Information Sciences Inst.; 1980 August 28; RFC 768. 3 p. (NIC.DDN.MIL RFC:RFC768.TXT).

Postel, J.B. Internet Protocol. Marina del Rey, CA: University of Southern California, Information Sciences Inst.; 1981 September; RFC 791. 45 p. (NIC.DDN.MIL RFC:RFC791.TXT).

Reynolds, J.K.; Postel, J.B. Assigned numbers. Marina del Rey, CA: University of Southern California, Information Sciences Inst.; 1987 May; RFC 1010. 44 p. (NIC.DDN.MIL RFC:RFC1010.TXT).

Obsoletes: RFC 990 (NACC 2139-86)

Reynolds, J.K.; Postel, J.B. Official Internet protocols. Marina del Rey, CA: University of Southern California, Information Sciences Inst.; 1987 May; RFC

1011. 52 p. (NIC.DDN.MIL RFC:RFC1011.TXT).

Romano, S.; Stahl, M.K.; Recker, M. Internet numbers. Menlo Park, CA: SRI International, DDN Network Information Center; 1988 August; RFC 1062. 65 p. (NIC.DDN.MIL RFC:RFC1062.TXT).

Internet Domain Registration Form

Note: The key people must have electronic mailboxes and NIC "handles," unique NIC database identifiers. If you have access to "WHOIS", please check to see if you are registered and if so, make sure the information is current. Include only your handle and any changes (if any) that need to be made in your entry. If you do not have access to "WHOIS", please provide all the information indicated and a NIC handle will be assigned.

To establish a domain, the following information must be sent to the NIC Domain Registrar (HOSTMASTER@NIC.DDN.MIL). Questions may be addressed to the NIC Hostmaster by electronic mail at the above address, or by phone at (415) 859-3695 or (800) 235-3155.

- (1) The name of the top-level domain to join.

Example:

COM

- (2) The NIC handle of the administrative head of the organization. Alternately, the person's name, title, mailing address, phone number, organization, and network mailbox. This is the contact point for administrative and policy questions about the domain. In the case of a research project, this should be the principal investigator.

Example:

Administrator

Organization	The NetWorthy Corporation
Name	Penelope Q. Sassafrass
Title	President
Mail Address	The NetWorthy Corporation 4676 Andrews Way, Suite 100 Santa Clara, CA 94302-1212
Phone Number	(415) 123-4567
Net Mailbox	Sassafrass@ECHO.TNC.COM
NIC Handle	PQS

- (3) The NIC handle of the technical contact for the domain. Alternately, the person's name, title, mailing address, phone number, organization, and network mailbox. This is the contact point for problems concerning the domain or zone, as well as for updating information about the domain or zone.

Example:

Technical and Zone Contact

Organization	The NetWorthy Corporation
Name	Ansel A. Aardvark
Title	Executive Director
Mail Address	The NetWorthy Corporation 4676 Andrews Way, Suite 100 Santa Clara, CA. 94302-1212
Phone Number	(415) 123-6789
Net Mailbox	Aardvark@ECHO.TNC.COM
NIC Handle	AAA2

- (4) The name of the domain (up to 12 characters). This is the name that will be used in tables and lists associating the domain with the domain server addresses. [While, from a technical standpoint, domain names can be quite long (programmers beware), shorter names are easier for people to cope with.]

Example:

TNC

- (5) A description of the servers that provide the domain service for translating names to addresses for hosts in this domain, and the date they will be operational.

A good way to answer this question is to say "Our server is supplied by person or company X and does whatever their standard issue server does."

Example:

Our server is a copy of the one operated by the NIC; it will be installed and made operational on 1 November 1987.

- (6) Domains must provide at least two independent servers for the domain. Establishing the servers in physically separate locations and on different PSNs is strongly recommended. A description of the primary and secondary server machines, including
- Host domain name and network addresses
 - Any domain-style nicknames (please limit your domain-style nickname request, if any, to one)
 - Hardware and software, using keywords from the Assigned Numbers RFC.

The preferred format for this information is:

Primary Server: HOST-DOMAIN-NAME, NETADDRESS, HARDWARE,
SOFTWARE

Secondary Server: HOST-DOMAIN-NAME, NETADDRESS, HARDWARE, SOFTWARE

Example:

Primary Server: BAR.FOO.COM, 10.9.0.13, VAX-11/750, UNIX

Secondary Server: XYZ.ABC.COM, 128.4.2.1, IBM-PC, MS-DOS

- (7) Planned mapping of names of any other network hosts (including any ARPANET or MILNET hosts), other than the server machines, into the new domain's naming space.

Example:

BAR-FOO2.ARPA (10.8.0.193) -> FOO2.BAR.COM
 BAR-FOO3.ARPA (10.7.0.193) -> FOO3.BAR.COM
 BAR-FOO4.ARPA (10.6.0.193) -> FOO4.BAR.COM

- (8) An estimate of the number of hosts that will be in the domain.

- (a) Initially
- (b) Within one year
- (c) Two years
- (d) Five years.

Example:

(a) Initially = 50
 (b) One year = 100
 (c) Two years = 200
 (d) Five years = 500

Note: Registration of a domain does not imply an automatic name change to previously registered ARPANET or MILNET hosts that will be included in the domain. Please list below the official host names and network addresses of any ARPANET or MILNET hosts that now appear in HOSTS.TXT whose names will change as a result of this domain registration. (Also be sure to answer question # 7, above.)

- (9) The date you expect the fully qualified domain name to become the official host name in HOSTS.TXT, if applicable.

- (10) Please describe your organization briefly.

Example:

The NetWorthy Corporation is a consulting organization of people working with UNIX and the C language in an electronic networking environment. It sponsors two technical conferences annually and distributes a bimonthly newsletter.

PLEASE ALLOW AT LEAST 10 WORKING DAYS FOR PROCESSING THIS APPLICATION

References

- [Birrell82] Birrell, A. D., Levin, R., Needham, R. M., and Schroeder, M. D., "Grapevine: An Exercise in Distributed Computing." In *Comm. A.C.M.* 25, 4, April 82.
- [Borden79] Borden, S., Gaines, R. S., and Shapiro, N. Z., *The MH Message Handling System: Users' Manual*. R-2367-PAF. Rand Corporation. October 1979.
- [Crocker77a] Crocker, D. H., Vittal, J. J., Pogran, K. T., and Henderson, D. A. Jr., *Standard for the Format of ARPA Network Text Messages*. RFC 733, NIC 41952. In [Feinler78]. November 1977.
- [Crocker77b] Crocker, D. H., *Framework and Functions of the MS Personal Message System*. R-2134-ARPA, Rand Corporation, Santa Monica, California. 1977.
- [Crocker79] Crocker, D. H., Szurkowski, E. S., and Farber, D. J., *An Internetwork Memo Distribution Facility — MMDF*. 6th Data Communication Symposium, Asilomar. November 1979.
- [Crocker82] Crocker, D. H., *Standard for the Format of Arpa Internet Text Messages*. RFC 822. Network Information Center, SRI International, Menlo Park, California. August 1982.
- [Metcalf76] Metcalfe, R., and Boggs, D., "Ethernet: Distributed Packet Switching for Local Computer Networks," *Communications of the ACM* 19, 7. July 1976.
- [Feinler78] Feinler, E., and Postel, J. (eds.), *ARPANET Protocol Handbook*. NIC 7104, Network Information Center, SRI International, Menlo Park, California. 1978.
- [NBS80] National Bureau of Standards, *Specification of a Draft Message Format Standard*. Report No. ICST/CBOS 80-2. October 1980.
- [Neigus73] Neigus, N., *File Transfer Protocol for the ARPA Network*. RFC 542, NIC 17759. In [Feinler78]. August, 1973.
- [Nowitz78a] Nowitz, D. A., and Lesk, M. E., *A Dial-Up Network of UNIX Systems*. Bell Laboratories. In UNIX Programmer's Manual,

- Seventh Edition, Volume 2. August, 1978.
- [Nowitz78b] Nowitz, D. A., *Uucp Implementation Description*. Bell Laboratories. In UNIX Programmer's Manual, Seventh Edition, Volume 2. October, 1978.
- [Postel74] Postel, J., and Neigus, N., Revised FTP Reply Codes. RFC 640, NIC 30843. In [Feinler78]. June, 1974.
- [Postel77] Postel, J., *Mail Protocol*. NIC 29588. In [Feinler78]. November 1977.
- [Postel79a] Postel, J., *Internet Message Protocol*. RFC 753, IEN 85. Network Information Center, SRI International, Menlo Park, California. March 1979.
- [Postel79b] Postel, J. B., *An Internetwork Message Structure*. In *Proceedings of the Sixth Data Communications Symposium*, IEEE. New York. November 1979.
- [Postel80] Postel, J. B., *A Structured Format for Transmission of Multi-Media Documents*. RFC 767. Network Information Center, SRI International, Menlo Park, California. August 1980.
- [Postel82] Postel, J. B., *Simple Mail Transfer Protocol*. RFC821 (obsoleting RFC788). Network Information Center, SRI International, Menlo Park, California. August 1982.
- [Schmidt79] Schmidt, E., *An Introduction to the Berkeley Network*. University of California, Berkeley California. 1979.
- [Shoens79] Shoens, K., *Mail Reference Manual*. University of California, Berkeley. In UNIX Programmer's Manual, Seventh Edition, Volume 2C. December 1979.
- [Sluizer81] Sluizer, S., and Postel, J. B., *Mail Transfer Protocol*. RFC 780. Network Information Center, SRI International, Menlo Park, California. May 1981.
- [Solomon81] Solomon, M., Landweber, L., and Neuhengen, D., "The Design of the CSNET Name Server." CS-DN-2, University of Wisconsin, Madison. November 1981.
- [Su82] Su, Zaw-Sing, and Postel, Jon, *The Domain Naming Convention for Internet User Applications*. RFC819. Network Information Center, SRI International, Menlo Park, California. August 1982.
- [UNIX83] *The UNIX Programmer's Manual, Seventh Edition, Virtual VAX-11 Version, Volume 1*. Bell Laboratories, modified by the University of California, Berkeley, California. March, 1983.

Index

A

- absolute mode, 139
- access permissions, 135
- accounting programs, 197 *thru* 217
- acctcom — examine accounting records, 213
- add_client — add NFS client, 173 *thru* 178
- add_services — add software, 178 *thru* 185
- adding hardware to system, 315 *thru* 344
- Address Resolution Protocol, *see* ARP under protocols
- administering a system
 - basic tasks, 5
 - developing procedures, 8
 - files and databases, 37 *thru* 49
 - local vs. network, 37
- administering electronic mail, 635 *thru* 649
- administering RFS, 555 *thru* 610
- administering secure NFS, 426
- administering workstations, 169 *thru* 217
- administrator awareness of security issues, 166
- administrator's guide to security, 149 *thru* 167
- adv — advertise RFS resources, 574
- /etc/aliases file, 644
 - overview, 41
 - protecting, 164
- application gateway, **380**
- application layer, 365
- architecture
 - application, 25
 - definition, 57
 - kernel, 25
 - table of kernel architectures, 3
- architecture pair, 25
- archiving
 - and the dump command, 91
 - planning a strategy, 92
- ARP, 363
- ARPANET, 54
 - glossary definition, 57
- asynchronous serial ports
 - connecting devices to, 320 *thru* 322
- at — do job at specified time, 157
- audit — maintain audit trail, 617, 835
- audit trail, 633
- audit_control file
 - sample, 630
- auditing, 616
- auditing, *continued*
 - audit file server, 622
 - audit file system, 618, 622
 - audit flag, 618
 - audit flags listing, 624
 - audit records format summary, 835 *thru* 852
 - audit state change, 617
 - audit trail administration, 611
 - audit trail examination, 633
 - audit value definition, 618
 - C2 requirements, 831
 - C2 security administration, 611
 - changing security parameters, 631 *thru* 633
 - event class, 617
 - immediate change, 617
 - permanent change, 617
 - process audit state, 617
 - superuser activities, 832
 - system audit value, 616
 - terminology, 616
 - user audit value, 616
 - vs. time and disk space, 832
- authentication
 - and break-ins, 434
 - DES, 429
 - DES applications, 433
 - and naming network entities, 432
 - performance, 435
 - RPC, 429
 - UNIX, 429
- authoritative data, 514
- Automatic Call Unit, 651, 656
- automount, 439 *thru* 468
 - and symbolic links, 440, 463
 - debugging, 461
 - and diskless clients, 439
 - vs. mount, 439
 - mount table, 460
 - obtaining process ID, 460
 - starting, 457
 - synopsis, 439, 462
- automount maps, 440
 - direct, **442**, 446 *thru* 451
 - indirect, **442**, 445 *thru* 446
 - master, **441**, 443 *thru* 444
 - modifying, 460
 - and escapes with special characters, 454
 - and environment variables, 455

automount maps, *continued*
 specifying subdirectories, 451
 and use of &, 452
 and use of *, 453

B

backing up file systems, 58
 using dump command, 89 *thru* 92
 dump strategies, 81
 on standalone system, 80
 over the network, 99
 procedures, 92 *thru* 101
 server files, 80
 tape specifications, 87 *thru* 89

backing up servers
 scheduling, 83

bar — create tape archives, 92

Berkeley Internet Name Domain service, *see* BIND

/bin directory, 20

BIND, 514

binding, **399**
 glossary definition, 57
 NIS, 472

block
 checking free blocks, 761
 fragments in, 759
 size, 758
 superblock, 757

/boot file, 20

boot block, 69
 glossary definition, 57

boot file, 521
 file: hosts, 524
 file: localhost, 524
 file: reverse hosts, 524
 file: root servers, 523
 for a forwarding server, 526
 for a primary and secondary server, 525
 for a primary server, 522
 for a secondary server, 525
 \$INCLUDE directive, 523
 type: cache, 523
 type: directory, 522
 type: forwarders, 526
 type: primary, 524
 type: slave, 526
 type:secondary, 525

booting
 aborting, 73
 from an alternate default device kernel, 804
 from alternative disk, 801
 automatic, 69 *thru* 73
 b command, 69
 boot command syntax, 74
 boot file, 69
 and the /etc/bootparams file, 71
 changes in Release 4.1, 67
 from devices, 73
 and diskless machines, 436
 error messages during, 805 *thru* 823
 and the fastboot command, 77
 and the fasthalt command, 77
 from tape, 804

booting, *continued*
 glossary definition, 57
 and the halt command, 74, 76
 and the init daemon, 75
 installboot program, 69
 kernel, 12, 70
 from a local disk, 69
 and the mount command, 70
 from an NFS-mounted partition, 803
 over the network, 70
 procedure for new clients, 70
 and the rc script, 76
 and rc.local, 75
 and the reboot command, 77
 and the rpc.mountd daemon, 72
 self-test procedure, 68
 and setuid, 436
 in single-user mode, 74, 436
 syntax, 74
 TCP/IP, 371
 and the TFTP(Trivial File Transfer Protocol) procedure, 70
 troubleshooting problems with, 436
 vmunix, 70

bootparamd daemon, 71

/etc/bootparams file
 and booting, 71
 overview, 46

breach of security, 167

bridge, **380**

bus
 glossary definition, 57

C

C. — work file, 678

C2 security requirements, 829
 auditing, 831
 auditing superuser activities, 832
 auditing vs. time and disk space, 832
 and discretionary access control, 830
 glossary, 829
 identification and authentication, 831
 and object reuse, 830
 and system architecture, 833
 and system integrity, 834

C2conv — convert to C2 security, **618**, 613
 sample, 620 *thru* 627

C2conv_input file, 627

C2conv_script, 627

C2unconv — convert from C2 security, 634

caching-only server, 526

catman — create cat files for manual pages, 16

changing a user's group ID, 135

changing security parameters, 631
 audit file, 632
 file permission modes, 140
 owner and group, 142
 system audit state, 631
 user audit state, 631

chat script, 660

check quotas — quotacheck, **122**

chgrp — change group ownership, 142

chkey — change encryption key, 427, 479

chmod — change mode, 137, 140
 chown — change owner, 141, 146
 client
 administering, 52
 dataless, 4, 58
 dataless vs. diskless, 18
 diskless, 4, 58
 file systems, 18
 (NFS) and the `fstab` file, 406
 and mounting, 399
 NFS, 398
 (NFS) setting up and maintaining, 405
 synchronizing time with server, 425
 types, 54
 communications
 domains for mail routing, 637
 communications controller status, 391
 computer viruses, 153
 computer worms, 153
`/usr/etc/config` — configuration-build utility, 220, 693 *thru*
 700
 default specifications, 696
 configuring kernel, 693 *thru* 700
 control entries
 `$INCLUDE`, 528
 `$ORIGIN`, 528
 controller
 disk controller device abbreviations, 91, 318
 Ethernet controller device abbreviations, 319
 glossary definition, 58
 graphics controller device abbreviations, 319
 tape controller device abbreviations, 86 *thru* 89, 318
 core image, 708
 cpio — copy file archives, 92
 crash — crash information, 707 *thru* 711
 crash dumps, 187
 crashes, 186 *thru* 190
 cron, 37, 164, 653
 protecting, 164
 crontab — editor, 192
 overview, 37
 using with `ypxfr`, 487
 crontabs file, 164
 crypt — encrypt, 142, 152
`.cshrc` file, 154
 cu — telephone connection to a remote system, 157
 cylinder groups, 758

D

D. — data file, 678
 daemon, 12
 glossary definition, 58
 `lockd` lock, 160
 `rpc.mountd`, 414
 `ypupdated` daemon, 427
 daemons
 `bootparamd`, 71
 `cron`, 37, 164
 `in.fingerd`, 366, 379
 `in.ftpd`, 365
 `in.named`, 513, 518
 daemons, *continued*
 `in.rlogind`, 366
 `in.rwhod`, 379
 `in.telnetd`, 365
 `in.tftpd`, 365
 `inetd`, 366
 `keyser`, 427, 432
 `lpd`, 12, 346, 358
 `named`, 365
 `nfsd`, 398, 400, 413, 422
 `portmap`, 505
 `pwdauthd`, 833
 `RARPD`, 71
 `rpc.mountd`, 398
 `rfudaemon`, 609
 `routed`, 381, 392
 `rpc.mountd`, 413, 415
 `rwhod`, 392
 `statd` status, 160
 `syslogd`, 193
 `uucico`, 653
 `uuxqt`, 653
 `uusched`, 654
 `ypbind`, 419, 470, 472, 473
 `yppasswd`, 628
 `ypserv`, 470, 472, 507
 `rpc.yppupdated`, 470, 479
 data
 authoritative, 526
 authoritative vs. cached, 514
 data link layer, 363
 database browsing, 548
 datagram, 364
 glossary definition, 58
 dataless client, 4, 18
 administering, 6
 files for administering, 42
 glossary definition, 58
 debugging
 and automount, 464 *thru* 468
 and UUCP, 683
 using `crash` on kernel images, 707 *thru* 711
 DNS, 549
 named daemon, 548
 peripheral add-ons, 322
 `sendmail`, 728
 server problems, 416 *thru* 418
`decryptsessionkey()` — decrypt session key, 432
 DES, 143
 authentication, 429
`/dev` directory, 20, 155
 device, 58
 adding, 323
 connecting to asynchronous serial ports, 320 *thru* 322
 device abbreviation
 disk controller, 91, 318
 Ethernet controller, 319
 graphics controller, 319
 miscellaneous boards, 319
 printer, 319
 tape controller, 86 *thru* 89, 318
 terminal multiplexor, 318
 device abbreviations, 18

device driver, 12
 adding, 694
 and adding boards, 315 *thru* 320
 glossary definition, 58
 pseudo-device, 62

device nodes, 20

device security, 155

device transfer speed, 665

device transmission protocols, 660

devices file, 656 *thru* 660, 693

df — report free disk space, **119**

dial-code abbreviations, 668

Dialcodes file, 668

Dialers, 660 *thru* 663

directories and set group ID, 147

disable — disable print queue, 355

disable quotas — `quotaoff`, **122**

discretionary access control, 135

disk
 boot block, 57
 cylinder groups, 758
 head, 59
 inode, 59
 label, 60
 maintenance, 249 *thru* 313
 partition, 61
 platter, 61
 sector, 62
 superblock, 62
 track, 63

disk quotas for file systems, 120 *thru* 124

`edquota` command, **122**

`quotacheck` command, **122**

`quotaoff` command, **122**

`quotaon` command, **122**

`repquota` command, **122**

diskless client, **4**, 18
 administering, 5
 and automount, 439
 files for administering, 37
 glossary definition, 58

DNS, **513**, 58, 365
 administering, 513 *thru* 554
 and adding a cache-only server, 545
 bibliography, 553
 caching-only servers, 520
 client-only vs. server/client, 519
 debugging, 549
 forwarding servers, 520
 and name space, 517
 and NIS, 499
 running without Internet, 545
 sample, 536
 server files, 520
 setting up, 547
 special characters, 527

DNS servers
 master server, 519
 slave server, 520

domain, *see* network
 mail routing, 637

domain administrator, 516, 550

Domain Name Service, *see* DNS

`domainname` — set/display domain name, 370, 484

`dorfs` — initialize RFS, 567

driver, *see* device driver

`du` — display disk blocks, **116**

`dump` — incremental file system dump, 58, 79, 91, 101
 full, **81**
 full, to 1/2 inch tape, 93 *thru* 95
 full, to 1/4 inch tape, 96 *thru* 98
 incremental, **81**
 incremental, to 1/2 inch tape, 95
 incremental, to 1/4 inch tape, 98
 syntax, 89
 and tape specs, 87, 88
 vs. `tar`, 79
 /usr, backing up to 1/2 inch tape, 94

`dumpdates` file
 determining when files were dumped, 101
 overview, 44

E

echo checking, 662

`edquota` — set disk quotas, **122**

EGP, 59

electronic mail, *see* mail, `sendmail`

`enable` — enable print queue, 355

enable quotas — `quotaon`, **122**

encryption, 142, *see also* `crypt` command
 DES, 143
 performance, 435
 public key, 431

`encryptsessionkey()` — encrypt session, 432

error messages
 from monitor and boot program, 805 *thru* 823

/etc directory, 20

/etc/hosts.equiv, 428

/etc/security/passwd.adjunct, 831

Ethernet, **54**
 and TCP/IP, 363
 backing up file systems over, 99
 glossary definition, 59
 and the `ifconfig` command, 388
 and repeaters, 380
 thin wire, 54

ethers database, 373
 overview, 48

/export/exec directory, 27

execute files, *see* `X.file`

/export directory, 20, 27 *thru* 31
 glossary definition, 59

`exportfs` — export directories to NFS, 398, 399, 403, 411, 413
 and `-u` option, **404**

/etc/exports file, **401**, 428, 434
 modifying, 402
 overview, 46

.exrc file, 154

Exterior Gateway Protocol, *see* EGP

`extract` command, 106

F

- fastboot command, *see* booting
- fasthalt command, *see* booting
- file access permissions, 135
- file creation mask (`umask`), 142
- file server, *see* server
- file sharing, 171
- File Transfer Protocol, *see* ftp
- files file, 693
 - backing up, *see* file system
 - restoring, *see* file system
- files.cmn file, 693
- filesystem, 12
 - back up procedures, 92 *thru* 101
 - backing up, 58, 79 *thru* 101
 - checking connectivity in, 763
 - client, 18
 - corruption, 760
 - edquota command, 122
 - /etc directory in 4.1, 32, 32, 34, 34
 - /export directory in 4.1, 27 *thru* 31
 - exporting via NFS, 398
 - for system administration, 37 *thru* 49
 - glossary definition, 59
 - hard-mounted NFS hierarchies, 416
 - High Sierra, 34
 - how to create hierarchies with NFS, 396
 - loopback, 34
 - making space on, 119
 - modifying for add-on printer, 344
 - modifying for hardware addition, 315 *thru* 344
 - monitoring usage, 115 *thru* 120
 - quotacheck command, 122
 - quotaoff command, 122
 - quotaon command, 122
 - quotas, 120 *thru* 124
 - recovering specific files, 103
 - repquota command, 122
 - restoring, 101 *thru* 115
 - restoring damaged root, 112
 - restoring entire, 108
 - root in 4.1, 20 *thru* 21
 - and servers, 18
 - soft-mounted NFS hierarchies, 416
 - superblock, 757
 - symbolic links in 4.1 file systems, 19
 - tmpfs, 34
 - updating, 759
 - /usr, 22 *thru* 27
- filesystem security, 155
- find — find files, 117
- format program, 249, 60
 - and changing/setting up disk partitions, 262
 - and creating defect list, 270
 - error messages, 301 *thru* 313
 - and formatting new Sun disk, 254, 257
 - and the `format.dat` file, 273 *thru* 278
 - invoking, 253
 - and the analyze menu, 289 *thru* 293
 - and the command menu, 278 *thru* 285
 - and the defect menu, 293 *thru* 301
 - and the partition menu, 286 *thru* 288
- format program, *continued*
 - overview, 249 *thru* 251
 - and relabeling corrupted disk, 268
 - and repairing sector, 264
 - user features, 251 *thru* 253
- forwarder, 380
- forwarding server, 526
- fragments in blocks, 759
- fsck — file system check program, 149, 186, 757 *thru* 787
 - checking connectivity, 763
 - checking free blocks, 761
 - checking inode data, 762
 - checking inode data size, 762
 - checking inode links, 761
 - checking inode states, 761
 - checking superblock, 760
 - error messages, 763 *thru* 786
- fstab file, 406 *thru* 408, 428
 - and clients, 38
 - modifying, 405, 407
 - server, 47
- ftp — file transfer program, 365
- fully-qualified domain names, 517

G

- gateway, 380
 - glossary definition, 59
- global file
 - glossary definition, 59
- glossary of C2 security terminology, 829
- glossary of terms, 57
- /etc/group file, 133, 135, 615
 - and NIS, 478
 - overview, 39
 - protecting, 164
- group ID, 145
- group membership, 134
- group.adjunct file, 616
- groups — group file format overview, 134
- grpuid option to mount, 147
- guest accounts, 150

H

- halt — stop the processor, 74, 76
- hardware
 - adding to system, 315 *thru* 344
- Hayes Smartmodem
 - adding to system, 327
 - and the `tip` command, 337
- head, 59
- heterogeneous server, 4
 - glossary definition, 59
- High Sierra file system, 34
- /home directory, 20
- home directory
 - backing up on servers, 83
 - backing up on standalone system, 85
 - creating, 170
 - in 4.1, 31
- homogeneous server, 4
 - glossary definition, 59

host, **51**
 glossary definition, 59
 host number, 367
 hostname — display host name, 429
 hosts database, 372
 and automount, 443
 and NIS, 477
 overview, 40, 47
 hosts.equiv file, **376**, 154, 378
 overview, 40
 use of '+' in, 376
 hosts.rev file, 518
 hsf s, 34

I

ICMP, 365, 387
 idload — load RFS security maps, 600
 ifconfig — configure network interface parameters, 371, 386, 388
 immediate user audit state, 632
 IN-ADDR.ARPA domain, 518, 524, 533
 in.fingerd daemon, 366, 379
 in.ftpd daemon, 365
 in.named daemon, 513, 518
 in.rlogind daemon, 366
 in.routed daemon
 and logging network problems, 393
 in.rwhod daemon, 379
 in.telnetd daemon, 365
 in.tftpd daemon, 365
 inetd daemon, 366
 init daemon, 75
 killing, 93
 initialization files, 154
 inode, **757**
 checking inode data, 762
 checking inode data size, 762
 checking inode links, 761
 checking inode states, 761
 definition, 757
 glossary definition, 59
 install directory
 overview, 45
 installboot program, 69
 Inter-Process Communications, *see* IPC
 interface, hardware network
 glossary definition, 59
 Internet, **53**
 and ARPANET, 54
 DARPA, 58
 DDN, 58
 DoD, 58
 glossary definition, 59
 NSFnet backbone, 61
 TCP/IP, 63
 vs. TCP/IP, 362
 Internet address, *see* IP address
 Internet Control Message Protocol, *see* ICMP
 internet datagram, 364
 Internet Protocol, *see* IP

internetwork, **379**, 55
 creating, 381
 glossary definition, 59
 and logical networks, 380
 internetwork mail routing, 755, 864
 IP
 and network layer, 364
 glossary definition, 60
 IP address, **367**, 56
 creating, 369
 glossary definition, 60
 adding to hosts database, 372
 host number, 367
 localhost, 372
 net number, 61, 367
 obtaining a net number, 368
 and RARP, 62
 representation, 367
 subnet number, 367
 IP header, 364
 IP network number, 367
 glossary definition, 60
 IP number registration form, 853
 IPC
 message parameters, 702
 named pipe parameters, 704
 semaphore parameters, 702
 shared memory parameters, 703
 IPC extensions, System V, 700 *thru* 704

K

/kadb directory, 21
 kernel, 25
 architecture, 3
 architecture types, 57
 basic functions, 12
 and booting, 12
 booting vmunix, 70 *thru* 73
 configuration file, 220 *thru* 229
 configuring for additional terminals, 323
 configuring to add boards, 315
 device driver, 58
 examining crashed image of, 707 *thru* 711
 glossary definition, 60
 modification procedures, 229 *thru* 242
 modifying for add-on modem, 328
 modifying for add-on printer, 342
 page, 61
 process, 61
 profiled, 696
 kernel and security auditing, 615
 kernel configuring, 693 *thru* 700
 kernel reconfiguring, 219 *thru* 248
 procedures, 242
 keylogin — decrypt and store secret key, 428, 434
 keylogout — remove secret key, 428
 key serv daemon, 427, 432
 kmem group, 834
 /usr/kvm
 contents in 4.1, 25 *thru* 27

L

label
 glossary definition, 60

LAN, **52**
 glossary definition, 60
 remote system dumps on, 99
 restoring files on, 107

LCK — lock file, 678

ld.so — dynamic link editor, 26

ldconfig — link-editor configuration, 26

/lib directory, 21

libraries, **13**
 shared, 190

library routines
 glossary definition, 60

line printer system, 345

link count, 148

ln — make link, 148

local administrative domains, 516

local file
 glossary definition, 60

localhost, 524

localhost address, 372

Lock Manager, 424

lockd lock daemon, 160

lockscreen — save window context, 151, 155, 833

lofs, 34

login — log in to the system, 153, 154, 157, 165, 428, 434
 adding via UUCP, 680
 Systems file field, 666

loopback, 34

/lost+found directory, 21

lpc — line printer control, 358
 exiting, 354
 starting, 353

lpd daemon, 12, 346, 358

lpq — display print queue, 147, 357

lpr — print, 147, 356

lprm — remove print job, 147, 358

ls — list directory contents, **115**
 options, 154
 and displaying permissions, 140

M

machine security, 165

mail, 55, 157
 administering, 635 *thru* 649
 enabling for workstations, 172
 and the Internet Domain Name Server, 647
 mail system command list, 635
 postmaster alias, 644
 system log, 649
 and UUCP, 683

Mail Exchanger, 534

Mail Group Member, 536

mail routing, 755, 864

mail routing domain, 637

Mailbox Information, 535

mailhost, 640
 configuring, 642

.mailrc file, 154

maintenance
 backing up file systems, 89 *thru* 101
 configuring kernel, 693 *thru* 700
 crashes, 186 *thru* 190
 disks, 249 *thru* 313
 fixing NIS problems, 502 *thru* 508
 local modifications, 191
 of printers and print servers, 345 *thru* 358
 restoring file systems, 107
 RFS, 603 *thru* 607
 system logs, 193 *thru* 197
 UUCP scripts, 682

make — process target programs and files, 60

makedbm, 480

MAKEDEV — make system special files, 20, 113, 317

makefile
 glossary definition, 60

Makefile.src file, 693

man page, **14**

map
 glossary definition, 60
 NIS, **470**
 propagation of, 487

master name servers
 primary master server, 519
 secondary master server, 519

Maxuuscheds file, 677

Maxuuxqts file, 677

mkdir — make directory, 407

mknod — build special file, 329, 334

/mnt directory, 21

modem, **54**
 adding a Telebit TrailBlazer to system, 327
 adding a USRobotics Courier 2400 to system, 327
 adding Hayes Smartmodem to system, 327
 adding to system, 326, 339
 Automatic Call Unit, 651
 automatic dialing, 658, 659
 Dialer-Token-Pairs, 658
 glossary definition, 60
 null, 320
 setting characteristics, 663, 667
 transmission rates, 651
 and connecting UUCP, 681

monitor
 error messages from, 805 *thru* 823
 and self-test procedure, 68

mount — mount file systems, **408**, 159, 399, 411, 420
 changing fstab using, 412
 glossary definition, 60
 grpfd, 147
 NFS mount, 18, 406
 RFS mount, 18
 UFS mount, 110

mount(2) system call, 462

mount point, **407**, 16, 21
 /-, 443
 /home, 443
 /net, 443

mount table, 460

mounting, **399**, 16, 159, 406

- mounting, *continued*
 - 4.2 mount, 17
 - RFS resources, 580
 - mt — magnetic tape control, 87
 - mtab, **411**
 - forcing re-reading of, 460
 - multiple mounts, 447
 - and mount points, 448
 - mount point conflicts, 461
 - example, 449
 - hierarchical, 448
 - mount point locations, 450
 - mv — move files, 148
 - MX Resource Record
 - preference value, 534
- N**
- name, 55
 - Name Server, *see* NS
 - name space, 517
 - named daemon, 365
 - debugging, 548
 - PID, 548
 - starting, 547
 - /etc/named.boot file, 520
 - National Computer Security Center
 - security criteria, 612
 - ncheck — convert i-numbers to filenames, 158
 - net number, 367
 - glossary definition, 61
 - obtaining, 368
 - netgroup, 47
 - glossary definition, 60
 - overview, 48
 - netgroup database, 377
 - netid.byname map, 432
 - netmasks database, 386
 - netstat — display network status, 389 *thru* 392
 - network, **361, 51**
 - authentication and naming, 432
 - backing up, *see* file system
 - backing up files remotely on, 99
 - booting, 803
 - booting over, 70
 - Class A numbers, 367
 - Class B numbers, 368
 - Class C numbers, 368
 - configuration, 54
 - diagnosing problems with, 387 *thru* 393
 - domain, 5, 8, 55, 58, 369
 - domain name, 61
 - domain name selection, 369
 - domain zones, **516**, 63
 - domain, registering, 370
 - domain, root level, 515
 - domain, second level, 516
 - domain, selecting name for host, 370
 - domain, top level, 515
 - domain, top level .COM, 370
 - domain, top level .EDU, 370
 - domain, top level .GOV, 370
 - and Ethernet, 54
 - network, *continued*
 - expanding, 379 *thru* 387
 - host, 51
 - IN-ADDR.ARPA domain, 518
 - IN-ADDR.ARPA domain, 524, 533
 - Internet, 53, 59
 - internetwork, 59
 - LAN, 52, 60
 - and mounting, 399
 - NIS domain, 63
 - packet, **363**, 61, 380
 - restoring files over, 107
 - RFS domain, 62
 - RPC, 62
 - software, 51
 - subnet, 62
 - TCP/IP, 63
 - UUCP, 53
 - WAN, 53, 60
 - network address, 462
 - network databases
 - modifications for NIS, 476
 - network equipment
 - modem, 60
 - router, 62
 - Network File System Service, *see* NFS
 - network hardware, 379
 - gateway, 59
 - Network Information Center, *see* NIC. *See* SRI-NIC
 - Network Information Service, *see* NIS
 - network layer, 364
 - network masks, **384**
 - network number
 - and classes, 367
 - and subnets, 368
 - network protocols, **361**, 61
 - and application layer, 365
 - and Ethernet, 363
 - and network layer, 364
 - and Token Ring, 363
 - and transport layer, 364
 - ARP, 363
 - DNS, 365
 - ftp, 365
 - RARP, 363
 - TCP/IP, 362
 - telnet, 365
 - tftp, 365
 - network security, 376, 423 *thru* 437
 - and .rhosts file, 378
 - adding users via C2, 628
 - administering, 125, 149, 167, 426
 - administering C2, 611 *thru* 634
 - and the /etc/aliases file, 164
 - allowing access, 378
 - and administrative files, 377
 - and NIS maps, 489 *thru* 491
 - and SunOS Release 4.1 enhancements, 613
 - bibliography, 437
 - breach, 167
 - C2 terminology, 616
 - changing a user's group ID, 135
 - changing auditing parameters, 631 *thru* 633

- network security, *continued*
 - changing permission parameters, 140
 - converting password and group files, 615
 - converting from C2, 634
 - converting to C2, 613
 - and cron, 164
 - encryption, 431
 - and the `/etc/group` file, 133, 164
 - group membership, 134
 - and `hosts.equiv` file, 378
 - and kernel requirements for C2, 615
 - packet smashing, 425
 - promoting awareness of, 165
 - publickey map, 479, 469
 - `setgid`, 159
 - `setuid`, 159
 - summary, 437
 - tips, 151, 166
 - and workstations, 172
 - network services, 54
 - networks database, 373 *thru* 374
 - overview, 48
 - `newfs` — construct a new file system, 109
 - `newgrp` — change group id of user, 135, 157
 - `newkey` — create a key in publickey database, 427, 479
 - NFS, 395, 54
 - administering, 395 *thru* 437
 - binding, 57, 399
 - booting from mounted file system, 803
 - global file, 59
 - glossary definition, 61
 - local file, 60
 - and setting up mailbox server, clients, 638
 - mount, 18, 406
 - mount point, 407
 - mounting, 399
 - troubleshooting, 412 *thru* 423
 - NFS client, 398
 - adding and removing, 173 *thru* 185
 - and the `fstab` file, 406
 - and mounting, 399
 - setting up and maintaining, 405
 - NFS files
 - hard-mounted, 416
 - soft-mounted, 416
 - NFS security, 423 *thru* 437
 - administering, 426
 - bibliography, 437
 - encryption, 431
 - and secure mounts via `C2conv`, 623
 - packet smashing, 425
 - summary, 437
 - NFS server, 18
 - creating and maintaining, 399 *thru* 405
 - and exporting directories, 401
 - and mounting files, 406
 - upgrading from homo- to heterogeneous, 405
 - `nfsd` daemon, 398, 400, 413, 422
 - NIC, 551
 - glossary definition, 61
 - handle, 61
 - net number, 61
 - and SRI, 62
 - nickname, 533
 - NIS, 469, 469 *thru* 509
 - adding users to, 491 *thru* 492
 - administering, 489 *thru* 491
 - and changing passwords, 490
 - and Domain Name Service, 499
 - binding, 472, 57
 - domain, 471, 58, 63, 475
 - global file, 59
 - glossary definition, 63
 - local file, 60
 - and administering mail, 646
 - and `Makefile`, 480
 - map, 60, 63, 473, 480
 - setting up, 475 *thru* 489
 - troubleshooting, 502 *thru* 508
 - turning off, 508
 - NIS client, 485, 472
 - NIS commands
 - summary, 501
 - NIS daemons
 - `rpc.ypupdated`, 470
 - `ypbind`, 470, 472, 473
 - `ypserv`, 470, 472
 - `ypserv`, 507
 - `ypupdated`, 479
 - NIS domains
 - resolving problems with, 500
 - and C2 security, 615
 - NIS maps, 470, 469, 473 *thru* 480
 - administering, 493
 - and security, 489 *thru* 491
 - changing a master server, 498
 - getting info on, 492
 - listing, 509
 - location of, 473
 - Makefile modification guidelines, 494 *thru* 498
 - map types, 493
 - modifying, 485, 487
 - `netid.byname`, 432
 - propagating, 482, 487 *thru* 496
 - publickey, 433
 - publickey map, 479, 469
 - updating, 493 *thru* 496
 - `yppush` command, 487
 - `ypxfr` command, 487 *thru* 489
 - NIS server, 471 *thru* 472
 - debugging, 506
 - expanding number of, 497
 - master, 471, 482, 483, 498
 - slave, 471, 484 *thru* 485, 486
 - NS resource record, 531
 - NSFnet
 - glossary definition, 61
 - `nslookup` — query name servers interactively, 549, 551
 - null modem, 320
- ## O
- object modules
 - sharing, 696
 - octets, 369, 518
 - operating system, *see* kernel

Orange Book/C2 requirements, 829 *thru* 834

P

packet, *see* network
 page, *see* virtual memory
 setting parity, 663, 668
 partition
 glossary definition, 61
 /etc/passwd file, 128, 157, 161, 428, 615
 maintaining, 150
 modifying, 131
 and NIS, 130, 132, 477
 overview, 38
 protecting, 161
 passwd.adjunct file, 616
 modifying, 625
 password
 administering, 150
 changing root, 13
 password aging, 161
 periodic jobs table, *see* crontab
 permanent user audit state, 631
 Permissions file, 669 *thru* 675
 CALLBACK option, 672
 combining options, 675
 COMMANDS option, 672
 and forwarding, 675
 MACHINE option, 675
 MYNAME option, 671
 NOREAD and NOWRITE options, 672
 READ and WRITE options, 671
 REQUEST option, 670
 SENDFILES option, 670
 setting up, 680
 VALIDATE option, 673
 permissions for access, 135
 physical layer, 363
 physical security, 165
 ping — send echo to host, 387, 450
 platter, *see* disk
 Poll file, 676
 polling remote computers, 676
 port
 numbers, 61, 364
 port selector, 656, 666
 portmap daemon, 505
 ports
 privileged, 424
 postmaster alias, 644
 praudit — display audit trail, 633, 835
 preference value, 534
 printcap file, 345
 editing for new printers, 347 *thru* 350
 overview, 41
 printer
 adding to system, 340, 344
 hooking up to VPC-2200 Multibus Board, 342
 printer daemons
 aborting, 355
 stopping and starting, 355
 printer queues

printer queues, *continued*
 enabling, disabling, 355
 printing
 access control, 352
 administering, 353
 command summary, 353
 editing printcap file, 347 *thru* 350
 enabling for workstations, 172
 and the lpd daemon, 12
 maintenance, 345 *thru* 358
 and output filters, 350
 on parallel ports, 349
 restarting, 354
 on serial lines, 348
 setting up, 346
 privileged ports, 424
 process, 12
 glossary definition, 61
 .profile file, 154
 protocol
 displaying statistics of, 389
 protocols
 ARP, 57
 DNS, 365
 EGP, 59
 ftp, 365
 glossary definition, 61
 network, 361, 61
 RARP, 62
 TCP/IP, 362
 telnet, 365
 tftp, 365
 UDP, 63
 protocols database, 374
 pseudo-device
 glossary definition, 62
 public key encryption, 431
 publickey database, 426, 433
 publickey map, 479, 469
 publickey.byname, 431
 push, 487
 pwdauthd daemon, 833

Q

quot command — check block quotas, 117
 quotacheck command — check disk quotas, 122
 quotaoff command — turn off disk quotas, 122
 quotaon command — turn on disk quotas, 122
 quotas for file systems, 120 *thru* 124

R

RARP, 62, 363
 /etc/rc script, 21, 76
 rc.local script, 75, 398
 modifying, 628
 rdate — set system date from a remote host, 425
 rdump — incremental file system dump, 99
 reboot command, *see* booting
 reconfiguring the kernel, 219 *thru* 248
 relay, 380
 /etc/remote file, 337

- Remote File Sharing Service, *see* RFS
- remote procedure call, *see* RPC
- remote.unknown file, 677
- repeater, **380**, 62
- repquota — report disk quotas, 122
- /etc/resolv.conf file, 519, 520, 521
 - samples, 540
- resolver, 514, 519
 - configuration file, 521
 - starting, 547
- Resource Record, 526
 - MB, 534
 - MR, 535
 - A, 531
 - class, 527
 - CNAME, 533
 - control entries, 528
 - HINFO, 532
 - MG, 536
 - MINFO, 535
 - MX, 534
 - name, 526
 - NS, 531
 - PTR, 533
 - RR data, 527
 - special characters, 527
 - ttl, 526
 - type, 527
 - WKS, 532
- Resource Record types, 528
- restore — incremental file system restore, 102
- restoring damaged root file systems, 112
- restoring entire file systems, 108
 - procedures, 111
- restoring file systems
 - determining which files to restore, 101
 - over the network, 107
 - procedures for recovering individual files, 104 *thru* 106
 - recovering specific files, 103
- Reverse Address Resolution Protocol, *see* RARP
- RFCs (Request for Comment), 551
- rfmaster file, 565
- RFS, **556**, 54
 - and the adv command, 574
 - advertising resources on, 573, 576
 - advertising tables, 573
 - and aliases, 578
 - starting on client, 571
 - domain, 58, 62, 564
 - defining domain name, 565
 - and the dorfs command, 567
 - glossary definition, 62
 - installing, 562 *thru* 564
 - mounting resources, 580
 - setting up name server for, 564
 - overview, 556 *thru* 562
 - and the rfmaster file, 565
 - and the rstab file, 577
 - setting up secondary name server for, 568
 - setting up server for, 570
 - procedures for starting on server, 570
 - and the unadv command, 579
- RFS, *continued*
 - unmounting resources, 585
- RFS maintenance, 603
 - domain, 606
 - for server and client, 604
 - updating user and group information, 605, 607
- RFS mapping, 597 *thru* 603
- RFS resources, 555
 - advertising, 573, 579
 - devices, 576
 - maintaining, 578
 - mounting, 580
 - naming, 573
 - security, 578
 - unmounting, 585
- RFS security
 - procedures for client, 595
 - and domain, 590, 592
 - setting up for mount, 596
 - and the primary name server, 591
 - overview, 587 *thru* 590
 - maintaining password and group files, 592
 - setting up for user and group, 596
- rfuadmin script, 609
- rfudaemon, 609
- /.rhosts file, **376**, 154, 378
 - and NIS, 477
 - overview, 40
- rlogin — remote login, 428
 - and NFS, 425
 - and security, 378
- rm_client — remove client, 185
- root, *see also* superuser
 - password, 127
 - becoming superuser, 13
- root domain name servers, 515
- root file system
 - backing up to 1/4 inch tape, 97
 - contents in 4.1, 20 *thru* 21
- root level domain, 515
- root user name, *see* superuser
- /export/root directory, 31
- routed daemon, 381, 392
- router, **380**, 55
 - configuring, 382
 - glossary definition, 62
- routing
 - and network masks, 384
- routing table, 391
- RPC, **395**
 - authentication, 429
 - glossary definition, 62
- rpc.mountd daemon, 72, 398, 413, 414, 415
- rrestore — incremental file system restore, 107
- rstab file, 577
- runnacct — run daily accounting shell, 200 *thru* 203
 - billing users, 205
 - generating reports, 206 *thru* 213
 - troubleshooting, 203 *thru* 205
- rwhod daemon, 392

S

- savecore — save OS core dump, 697, 708
- /sbin directory, 21
- search path, 155
- second level domains, 516
- sector
 - glossary definition, 62
- secure keyword, 151
- security, 376
 - adding users via C2, 628
 - administering, 125, 149, 167, 426
 - administering C2, 611 *thru* 634
 - and the /etc/aliases file, 164
 - and administrative files, 377
 - and NIS maps, 489 *thru* 491
 - bibliography, 437
 - breach, 167
 - C2, **612**
 - C2 terminology, 616
 - changing a user's group ID, 135
 - changing auditing parameters, 631 *thru* 633
 - changing permission parameters, 140
 - computer viruses, 153
 - computer worms, 153
 - converting password and group files, 615
 - converting from C2, 634
 - converting to C2, 613
 - and cron, 164
 - data barriers on SunOS, 127
 - device, 155
 - encryption, 431
 - and SunOS Release 4.1 enhancements, 613
 - and file systems, 155
 - for users, 128 *thru* 149
 - and the /etc/group file, 133, 164
 - group membership, 134
 - guide for administrators, 149 *thru* 167
 - and kernel requirements for C2, 615
 - and NFS, 423 *thru* 437
 - and Orange Book/C2 requirements, 829 *thru* 834
 - overview, 125
 - packet smashing, 425
 - and the /etc/passwd file, 128, 161
 - and password aging, 161
 - UUCP Permissions file, 669 *thru* 675
 - physical, 165
 - port checking, 400
 - promoting awareness of, 165
 - protecting unattended workstations, 155
 - publickey map, 469, 479
 - RFS, 587 *thru* 603
 - and RFS resources, 578
 - summary of network, 437
 - system barriers on SunOS, 126
 - tips, 151, 153, 166
 - trojan horses, 152
 - trojan mules, 153
 - establishing workstation passwords, 171, 172
- /etc/security/audit directory, 618
- sendmail — mail delivery system, 755, 864
 - and domain names, 637
 - features, 714
 - installation and operation
 - sendmail — mail delivery system, *continued*
 - local alias database, 721, 727
 - mail queue, 725
 - overview, 715 *thru* 717
 - special header lines, 743
 - sendmail arguments
 - alternate configuration file, 728
 - daemon mode, 728
 - debugging, 728
 - queue interval, 728
 - sendmail configuration file
 - conditionals, 744
 - defining classes, 741
 - defining header, 743
 - defining macro, 741
 - defining mailer, 744
 - defining trusted users, 743
 - left hand side, 746
 - precedences, 742
 - rewriting rules, 743
 - right hand side, 747
 - setting option, 742
 - special classes, 746
 - special macros, 744
 - syntax, 734
 - sendmail installation
 - arguments, 728
 - basic installation, 719
 - command line flags, 751
 - configuration file, 732 *thru* 734
 - configuration options, 752 *thru* 753
 - mailer flags, 754 *thru* 755
 - normal operations, 719
 - support files, 719 *thru* 755
 - tuning, 729 *thru* 731
 - sendmail tuning
 - delivery mode, 730
 - file modes, 731
 - load limiting, 730
 - log level, 731
 - timeouts, 729
- serial ports
 - connecting devices to, 320 *thru* 322
- server, **398**
 - adding a cache-only server, 545
 - administering, 7, 52
 - and exporting file systems, 398
 - audit file server, 622
 - and authoritative vs. cached data, 514
 - back up schedule for, 83
 - and binding, 399
 - boot file for a forwarding server, 526
 - boot file for a primary and secondary server, 525
 - boot file for a primary server, 522
 - boot file for a secondary server, 525
 - caching-only, 58, 520
 - (NFS) debugging, 416 *thru* 418
 - default file system, 80
 - and exporting directories, 401
 - DNS files, 520
 - file, 4
 - file systems on, 18
 - files for administering, 43

- server, *continued*
- files to back up on, 80
 - forwarding, 520
 - glossary definition, 62
 - heterogeneous, 4, 59
 - homogeneous, 4, 59
 - kernel configuration, 316
 - mailbox, 638
 - master, 471, 482, 483, 498, 519
 - NFS, 18, 399 *thru* 405, 406
 - primary master, 61
 - RFS, 570
 - secondary master, 62
 - services provided by, 54
 - slave, 471, 484 *thru* 485, 486, 520
 - synchronizing time with client, 425
 - upgrading from homo- to heterogeneous via NFS, 405
- services database, 375
- set group ID, 146
- and directories, 147
 - and SunOS commands, 148
- set user ID, 145
- and SunOS commands, 148
- setgid bit, 145, 159
- setgid permission, 146
- setsecretkey () — store secret key, 432
- setuid bit, 145, 156, 159
- /export/share/sunos.4.1 directory, 31
- shared libraries, 190
- showfh — show file name, 412
- showfhd daemon, 412
- showmount — show all remote mounts, 411, 420
- shut down — close down the system, 76, 92, 167
- shutting down the system, 76 *thru* 77
- SIGINT, 548
- SIGUSR1, 549
- SOA, 530
- spooling, 345
- and output filters, 350
- SRI, 62
- SRI-NIC, 362, 368
- standalone system, 4, 18
- administering, 7, 52
 - backing up files on, 80
 - backup schedule for, 85
 - default file system on, 80
 - files for administering, 42
 - glossary definition, 62
- Standard Resource Record Format, 526
- Stanford Research Institute, International, *see* SRI
- Start of Authority, *see* SOA
- statd status daemon, 160
- sticky bit, 144
- stop — stop background job, 355
- su — become superuser, 14, 149, 151, 152, 157, 429, 434
- and NFS, 425
- subnet, 368
- creating, 384, 386
 - glossary definition, 62
 - number, 367
 - and ypbind daemon, 473
- suninstall command, 80
- SunOS commands and set user/group ID, 148
- SunOS Reference Manual*
- overview of, 14
- .sunview file, 154, 155
- superblock, 757
- checking, 760
 - glossary definition, 62
 - summary information, 758
- superuser, 13 *thru* 14, 62, 150
- changing root, 13
 - glossary definition, 63
- swap space, 18
- procedures for increasing, 246
 - swap generic, in kernel config file, 225
- /export/swap directory, 31
- swapon — specify paging device, 697
- symbolic link, 463
- definition, 63
 - in Release 4.1 file systems, 19
- synchronizing client-server time, 425
- /sys directory, 21
- Sysv file, 676
- syslog.conf file, 193
- syslogd daemon, 193
- system accounting programs, 197 *thru* 217
- system administration
- overview of files and databases, 37 *thru* 49
 - and printing, 353
- system configuration
- glossary definition, 58
- system logs, 193 *thru* 197
- system maintenance
- backing up file systems, 89 *thru* 101
 - configuring kernel, 693 *thru* 700
 - crashes, 186 *thru* 190
 - local modifications, 191
 - system logs, 193 *thru* 197
- System V
- IPC extensions, 700 *thru* 704
- SYSTEM_NAME file, 693
- Systems, 663 *thru* 668
- ## T
- tape
- 1/2 inch, backing up to, 93
 - 1/4 inch, backing up to, 95 *thru* 98
 - specifications for 1/2 inch, 87
 - specifications for 1/4 inch, 88
- tar command — create tape archives, 92
- vs. dump, 79
- TCP, 364
- glossary definition, 63
- TCP/IP, 53
- booting, 371
 - configuration options, 704 *thru* 707
 - Domain Name Service, 58
 - glossary definition, 63
 - hardware, 379
 - and IP address, 56
 - transmission unit, 58

- TCP/IP, *continued*
 - and virtual circuits, 63
 - vs. Internet, 362
 - TCP/IP protocols, 362
 - and Ethernet, 363
 - and Token Ring, 363
 - application layer, 365
 - ARP, 57
 - data link layer, 363
 - DNS, 365
 - ftp, 365
 - layers, 362
 - network layer, 364
 - physical layer, 363
 - telnet, 365
 - tftp, 365
 - transport layer, 364
 - technical contact, 551
 - Telebit TrailBlazer
 - adding to system, 327
 - telnet — interface to remote system, 365
 - temporary directories, 155
 - termcap database, **789**
 - area clears, 793
 - capabilities, 789, 791
 - cursor motions, 793
 - delays, 797
 - glitches, 799
 - highlighting, underlining, and visible bells, 795
 - insert/delete, 793
 - keypad, 796
 - modifying, 789 *thru* 800
 - tabs and initialization, 796
 - terminal
 - adding to system, 322, 326
 - tftp — trivial file transfer program, 365
 - /tftpboot directory, 21
 - tftpboot file, *see* booting
 - time-sharing system, *see* standalone system
 - timezone listing, 825
 - tip — terminal emulator, 157
 - Hayes Smartmodem, 337
 - tips for security, 153
 - TM — temporary data file, 677
 - /tmp directory, 21, 155
 - tmp_mnt, special directory, 440, 459
 - tmpfs, 34
 - token, 666
 - Token Ring
 - and TCP/IP, 363
 - top level domains, 515
 - track
 - glossary definition, 63
 - Transmission Control Protocol, *see* TCP, TCP/IP, 364
 - transport layer, 364
 - Trivial File Transfer Protocol, *see* tftp
 - trojan horses, 152
 - trojan mules, 153
 - troubleshooting
 - add-on boards, 320
 - add-on peripherals, 322
 - troubleshooting, *continued*
 - booting problems, 436
 - disk problems, 301
 - mail delivery problems, 648
 - network problems, 387 *thru* 393
 - NFS problems, 412 *thru* 423
 - NIS problems, 502 *thru* 508
 - printers and print servers, 356
 - RFS, 608 *thru* 610
 - system crashes, 707
 - and system log error messages, 195
 - terminal add-ons, 326
 - /etc/ttytab file, 127, 436
 - modifying, 151
 - modifying for modem add-ons, 330, 335
 - overview, 44
- ## U
- UDP, 364
 - glossary definition, 63
 - UDP socket, 462
 - umask — display file creation mask, 142
 - umount command, **410**, 109, 110, 159, 399, 451
 - and disabling quotas, 122
 - unadv — remove RFS resource from advertise tables, 579
 - UNIX
 - authentication, 429
 - user awareness of security issues, 165
 - User Datagram Protocol, *see* UDP
 - user ID, 145
 - user name
 - nobody, 423
 - user's guide to security, 128 *thru* 149
 - /usr, 21
 - /usr directories
 - /bin, 22
 - /dict, 23
 - /etc, 23
 - /games, 23
 - /hosts, 23
 - /include, 23
 - /kvm, 24
 - /lib, 24
 - /local, 24
 - /lost+found, 24
 - /pub, 24
 - /sccs, 24
 - /share, 24
 - /src, 24
 - /stand, 24
 - /sys, 24
 - System V directories, 23
 - /ucb, 25
 - usr file system
 - backing up to 1/4 inch tape, 96
 - contents in 4.1, 22 *thru* 27
 - USRobotics Courier 2400
 - adding to system, 327
 - UUCP, **53**, 651
 - and adding logins, 680
 - and connecting through a modem (UUCP), 681, 682
 - device protocols, 660

- UUCP, *continued*
 - device speed, 657
 - Dialer-Token-Pairs, 658
 - direct links, 651, 658
 - examples, 681
 - forwading through, 675
 - grades, 665
 - hardware, 651
 - hardware flow control, 663, 667
 - Honey DanBer, 651
 - LAN, 658
 - mail, 683
 - modem rates, 651
 - network, 652
 - parity setting, 663
 - running over TCP/IP, 679
 - and security issues, 669 *thru* 675
 - setting parity, 663
 - starting, 654
 - telephone lines, 651
 - transmission protocols, 660
 - as WAN, 53
 - UUCP administrative programs
 - uucheck, 653
 - ucleanup, 653
 - uulog, 653
 - Uutry, 653
 - UUCP daemons
 - uucico, 653, 663
 - uusched, 654, 677
 - uuxqt, 653, 677
 - UUCP debugging, 683
 - error messages, 684, 687
 - Systems file, 683
 - transmissions, 683
 - UUCP files
 - C. — work file, 678
 - correspondences, 668
 - D. — data file, 678
 - Devices, 655, 656 *thru* 660
 - Dialcodes, 655, 668
 - Dialers, 655, 660 *thru* 663
 - LCK — lock file, 678
 - Maxuuscheds file, 677
 - Maxuuxqts file, 677
 - Permissions, 669 *thru* 675, 680
 - Poll, 656, 676
 - remote.unknown file, 677
 - Sysfiles, 656
 - Sysfiles file, 676
 - Systems, 655, 663 *thru* 668
 - TM — temporary data file, 677
 - X.file, 679
 - UUCP maintenance, 682
 - uudemon.admin, 655
 - uudemon.cleanup, 655
 - uudemon.hour, 654
 - uudemon.poll, 654
 - UUCP user programs
 - cu, 652
 - uucp, 652
 - uupick, 652
 - uustat, 652
 - UUCP user programs, *continued*
 - uuto, 652
 - uux, 652, 672
 - UUCP version upgrade: differences, 685
 - uucp, 651, 675, 679
 - uudemon.poll, 676
- V**
- /var directory, 21
 - /tmp, 155
 - /var subdirectories, 21
 - /crash, 22
 - /preserve, 22
 - /spool, 21
 - /tmp, 22
 - /yp, 22
 - virtual circuit, 63, 364
 - virtual memory, 12
 - glossary definition, 63
 - page, 61
 - /vmunix directory, 21, *see* kernel, 147
 - VPC-2200 Multibus Board
 - hooking up for printer, 342
- W**
- WAN, 53
 - glossary definition, 60
 - Internet, 53, 59
 - UUCP, 53
 - what is — display keyword summary, 16
 - wide area network, *see* WAN
 - workstations
 - adding and removing, 173 *thru* 185
 - administering, 169 *thru* 217
 - creating home directory on, 170
 - enabling electronic mail, 172
 - enabling printing, 172
 - setting up file sharing, 171
 - modifying administrative databases for, 169 *thru* 172
 - and network security, 172
 - setting up password security, 171
 - setting up user groups, 172
- X**
- X.file, 653, 679
- Y**
- YP, *see* NIS
 - ypbind daemon, 419, 470, 472
 - and subnetting, 473
 - ypcat — print values from NIS database, 492
 - ypinit — make NIS database, 480, 482
 - ypmatch — match NIS keys, 493
 - yppasswd — change login password in NIS, 428, 432, 486
 - ypasswdd daemon, 628
 - yppush — force propagation of modified NIS map, 499
 - ypserv daemon, 470, 472
 - fixing crashes of, 507
 - ypupdated daemon, 427, 470, 479
 - ypwhich — who is NIS server, 473, 493
 - resolving problems with, 506

ypxfr — transfer NIS map, 487 *thru* 489
using with crontab, 487
and shell scripts, 488

Z

zones, 516, 63